
MSX
MAGAZINE
AÑO II
Núm. 17
Octubre
1986
300 Ptas.
Cómo ahorrar memoria
Test: Mitsubishi ML-G1 y ML-G3
Instrucciones ocultas del Z-80
Libros, programas, software
ROBOTS, TRABAJADORES INFATIGABLES
SVI

DIRECTOR:
Juan Arencibia.
COLABORADORES:
Octavio López, Angel Zarazaga,
Teresa Aranda, Ricardo García.
DISEÑO:
Benito Gil
Editada por:
PUBLINFORMATICA, S.A.
C/ Bravo Murillo, 377 - 5.º A
Tel.: 733 74 13
28020 Madrid.
Telex 48877 OPZXE
PRESIDENTE:
Fernando Bolín.
DIRECTOR EDITORIAL REVISTAS DE USUARIOS:
Juan Arencibia.
DIRECTOR DE VENTAS:
Antonio González.
JEFE DE PRODUCCIÓN:
Miguel Onieva.
SERVICIO AL CLIENTE:
Julia González.
Tel.: 733 79 69
DIRECCION, REDACCION Y ADMINISTRACÍON:
C/ Bravo Murillo, 377 - 5.º A.
Tel.: 733 74 13
28020 Madrid.
PUBLICIDAD EN MADRID:
Emilio García.
PUBLICIDAD EN BARCELONA:
Lidia Cendros.
C/ Pelayo, 12.
Tel.: (93) 301 47 00 Ext. 27-28.
08001 Barcelona.
Depósito Legal: M. 16.755-1985
Impreso en Héroes, S.A.
C/ Torrelara, 8. 28016 Madrid.
Distribuye:
S.G.E.L. Avda. Valdelaparra, s/n.
Alcobendas (Madrid).
DISTRIBUIDORES:
VENEZUELA: SIPAM, S.A.
Avda. República Dominicana, 541
ARGENTINA: DISTRIBUIDORA INTERCONTINENTAL BUENOS AIRES.

El P.V.P. para Ceuta, Melilla y Canarias, incluido servicio aéreo será de 300 ptas. sin I.V.A.

SUSCRIPCIONES:
Rogamos dirija toda la correspondencia relacionada con suscripciones a:
MSX
EDISA: Tel. 415 97 12
C/López de Hoyos, 141-5.º
28002 MADRID
(Para todos los pagos reseñar solamente MSX)
Para la compra de ejemplares atrasados diríjanse a la propia editorial
MSX
C/Bravo Murillo, 377-5.º A
Tel. 733 74 13 28020 MADRID

Si deseas colaborar en MSX remite tus artículos o programas a Bravo Murillo 377, 5.º A. 28020 Madrid. Los programas deberán estar grabados en cassette y los artículos mecanografiados.
A efectos de remuneración, se analiza cada colaboración aisladamente, estudiando su complejidad y calidad.

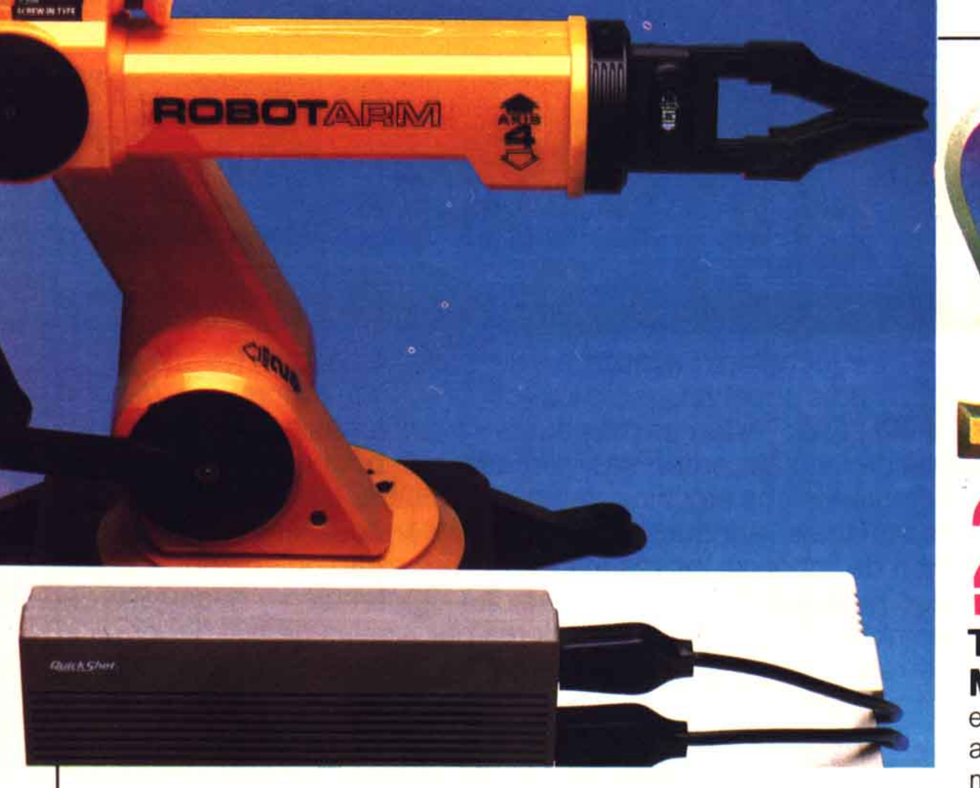


La tecnología avanza a pasos agingatados. Hoy día podemos observar, como los robots ayudan al hombre a desempeñar las tareas más duras y monótonas que hay. En cadenas de montaje, empresas químicas, etc., podemos ver como brazos mecánicos efectúan trabajos que hace una década eran inimaginables. Aunque estamos empezando la segunda generación de robótica, los resultados que se están obteniendo indican que no existe freno a la capacidad de creación y desarrollo del hombre. En lo que a ordenadores personales de la II generación se refiere, cabe destacar la aparición de un nuevo competidor: Mitsubishi.

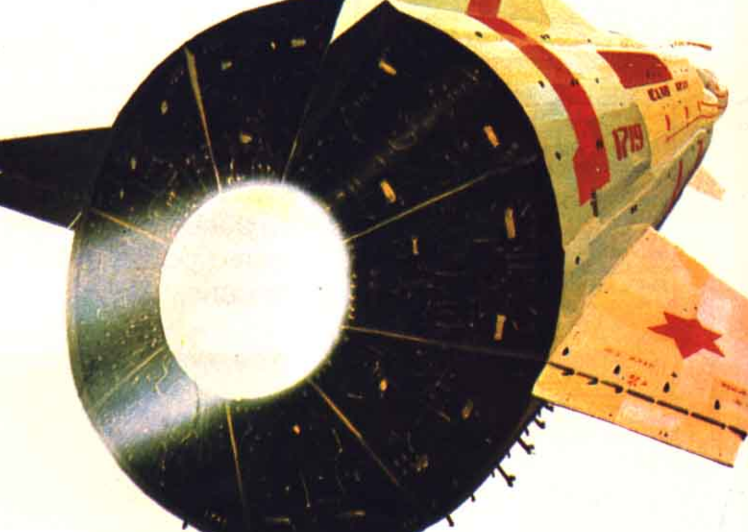
El mercado de los ordenadores de la II generación, estaba principalmente cubierto por dos empresas muy importantes como son Philips y Sony. Ambos, con el VG-8235 y HB-500P respectivamente, se repartían este nuevo sector. Sin embargo, esto no ha impedido a Mitsubishi plantarles cara en un terreno aún demasiado nuevo, con dos ordenadores que además de aumentar el cupo de aparatos de la II generación, presentan unas características poco usuales en estos ordenadores. Salvo el precio, todo está a favor de los nuevos equipos ML-G3 y ML-G1 (aunque parece ser que estos bajarán en un plazo breve), diseño, prestaciones, etc. Sólo la aceptación por parte del usuario será lo que marque las diferencias entre estos ordenadores. Este mes dedicamos la sección de Test, a estos nuevos ordenadores, dejando para más adelante una comparativa que englobe lo que vaya apareciendo en el mercado de los ordenadores de la II generación.

Por otro lado, SVI España distribuidor de los ordenadores Spectravídeo, se lanza a la aventura de los compatibles PC. Además de comercializar los ordenadores de la gama MSX (SVI-728 y 738 X'press) y los SVI-328 y 318, inicia una nueva etapa en el difícil mercado de los compatibles. Dichos aparatos van a tener un precio muy interesante y competitivo a todos los niveles. Desde 158.000 ptas. hasta 337.000 ptas. (según las opciones), lo que indica el importante giro que está tomando el mundillo de los ordenadores personales. Huelga decir, que una bajada de este tipo sólo puede desencadenar una guerra de precios, que ningún fabricante podrá evitar con el consiguiente beneficio por parte del usuario.





Su




mario


ML-G3
MSX

NOTICIAS

Nuevos sistemas de alimentación ininterrumpida

El eterno problema de las subidas y bajadas de tensión en la red está a punto de acabar, y todo gracias a los nuevos dispositivos de alimentación ininterrumpida que comercializa la empresa ITISA. De los cuatro sistems que posee, al usuario de ordenadores le interesará los dos sistemas más sencillos que son, un estabilizador de tensión y un sistema de alimentación ininterrumpida (los otros dos dispositivos son una alimentación ininterrumpida con *by pass* y un equipo estabilizador con *SAIS* y *by pass*).

Con el primero quedan eliminadas cualquier fluctuación de la tensión de la red, así como picos y variaciones bruscas de la misma.

El segundo dispositivo permite continuar utilizando el ordenador en ausencia de corriente, empleando la energía almacenada en la batería que el equipo lleva incorporada. Los otros dos dispositivos son más complicados y para ordenadores de más nivel.

De cualquier manera, para obtener más información, dirigirse a:

ITISA
Almansa, 10
28040 Madrid

Anaya, líder en publicación de libros

Continuando con la serie de libros publicados por esta editorial (que parece no tener techo), cabe resaltar la aparición de uno muy especial dedicado a la generación de música por ordenador. El libro, titulado «Proyectos de Muúsica con Microordenadores» de 152 páginas, muestra algunos de los modos en los uqe se puede utilizar un ordenador doméstico para la producción de música electrónica. El libro no está dirigido a principiantes, pero no es necesaria una gran comprensión del hardware y la programación, para constituir y utilizar los circuitos descritos.

Por otro lado, y continuando con la serie de libros que completan los manuales de importantes programas para ordenadores PC, pronto veremos los siguientes libros:

— «Técnicas Avanzadas en Framework»;
— «Programación en Pascal y Turbo Pascal»;
— «Trucos y Recursos en Symphony» y
— «Biblioteca del Programador en C».

Los prcios de estas publicaciones oxcilarán entre 2.650 y 6.990 ptas., según l libro ya que el último de estos (el más caro) lleva incorporado dos diskettes.

Para aprender Inglés en casa

Existen muchas formas de aprender Inglés, entre otras está la del manual acompañado de su cassette correspondiente, pero ninguno lo hace empleando las cualidades didácticas del ordenador.

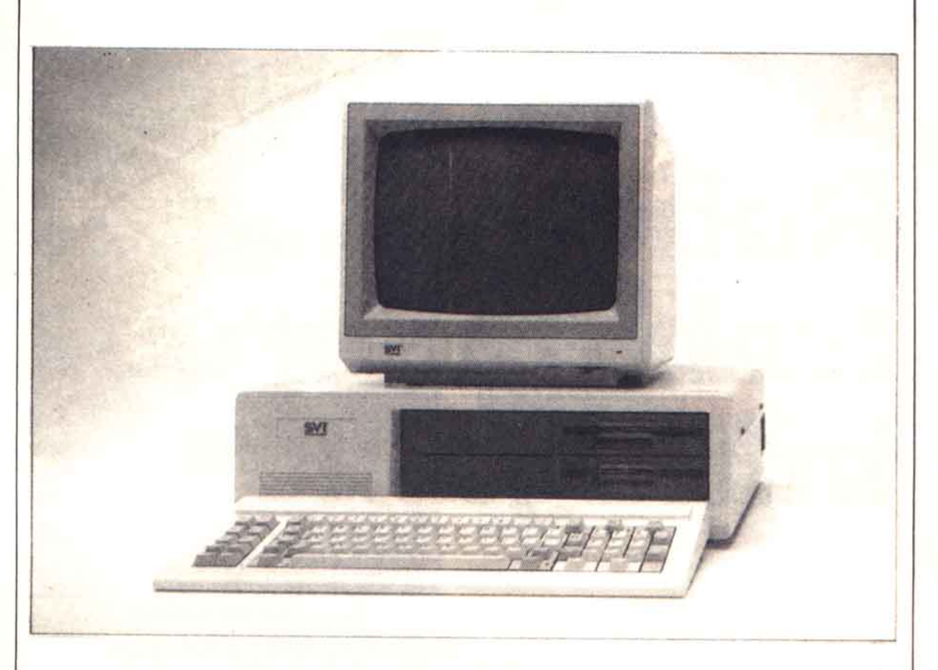
Actualmente, Plusdata S.A. comercializa un programa con el cual el usuario podrá iniciarse en el aprendizaje de este idioma tan importante. El sistema empleado es el Grunenberg Linkword Language System, desarrollado por el Dr. Michael Grunenberg, con el que en poco tiempo, unas 20 horas como máximo, el usuario aprenderá un vocabulario de unas 400 palabras.

Con este programa, Plusdata S.A. se incorpora en el creciente mercado de software didáctico, dando un paso muy importante en las posibilidades didácticas del ordenador. Para obtener más información, dirigirse a:

PlusDATA, S.A.
Gran Vía, 661
08010 Barcelona

Serma premia al Jefe de Compras de Galerias Preciados

Serma, importante distribuidor de software, ha hecho entrega al Sr. Thomas Enders (Jefe de Compras de Galerías Preciados) de una placa conmemorativa, premiando así el esfuerzo realizado en la introducción en el mercado español de sus productos. El Sr. Thomas Enders, ha realizado una labor muy importante en este sector potenciando la introducción de nuevos productos.



Nuevos compatibles PC de Spectravideo

Spectravideo inicia su andadura en el saturado mercado de los compatibles PC de IBM.

Continuando con su política de expansión, Spectravideo España, además de dedicarse a la distribución y venta de ordenadores MSX, empieza una nueva etapa.

Los equipos que se van a comercializar, son compatibles 100% (habrá que comprobarlo) con todos los programas existentes. Tres son los equipos que se van a comercializar, cuyas denominaciones son las siguientes: SVI-256 SF, SVI-640 FF y SVI-540 FH.

El primero de ellos tiene 256K de memoria RAM, ampliable a 640K y unidad de disco de 5 1/4". Saldrá al mercado a un precio de 158.000 ptas. (sin IVA).

El SVI-640 FF posee 640K de RAM y dos unidades de discos de 5 1/4". Su precio rondará las 188.000 ptas. (sin IVA).

Por último, el SVI-640 FH se comercializará con una memoria RAM de 640K, una unidad de disco de 5 1/4" y una unidad de disco duro Winchester de 20 Mb. Al sr el más completo de la gama, su precio es el más alto, 337.000 ptas. (sin IVA).

Además de estas características, posee también tarjetas de ampliación. Desde tarjetas de Gráficos en Color hasta tarjetas Multifunción, que incorpora interface RS-232C, reloj de tiempo real, port de juegos, zócalos para 2 Mb de RAM, software para el disco RAM y SPOOL de impresora. Para obtener más información, dirigirse a :

Prot SVI España
Avda. de la Constitución, 260
Torrejón de Ardoz (Madrid)

o a su delegación en Cataluña, en:

Avda. Pau Claris, 165, 3
08037 Barcelona

Robots, trabajadores infatigables

Robots, seres inanimados que hacen la labor del hombre, están muy distantes de competir con aquél en determinadas situaciones de la vida diaria, aunque su protagonismo en otros entornos, como son las fábricas e industria, son digno de la mejor novela de ciencia ficción.



Todavía estamos en la primera generación de la *robótica,* ciencia que estudia a los *robots* y su diseño. En ella es de destacar los brazos mecánicos, elementos que comienzan a sustituir al hombre en las tareas más tediosas y repetitivas.

Aún queda mucho tiempo para que los *robots* sean una realidad palpable en lugares públicos tales como tiendas, restaurantes, etc. Sin embargo, sus primeros pinitos los están realizando, con muy buenos resultados, en empresas de construcción, cadenas de montaje e incluso en algunos hogares.

Actualmente, mientras que por una parte nos encontramos en los inicios de los ordenadores de la quinta generación, sólo estamos empezando a recorrer el largo camino de la *robótica.* Como hemos dicho anteriormente, estamos finalizando la primera generación de *robots.* Estos se pueden definir como máquinas programa-



Foto 1. Detalle del Robot Arm de Spectravideo. Cinco ejes permiten una movilidad absoluta.

bles con poca o ninguna posibilidad sensorial o inteligencia. El principal ejemplo de tales dispositivos es el brazo mecánico, utilizado en fábricas y en cadenas de montajes de automóviles.

Hoy por hoy, son muchos los brazos mecánicos utilizados en las más diversas labores. Se pueden diferenciar por la fuente de energía que emplean (motores eléctricos, hidráulicos o neumáticos) y por la envolvente de trabajo.

Esta cualidad es el volumen de espacio que rodea al brazo donde puede trabajar con la pinza o «mano». La forma de la envolvente viene determinada por la construcción física del brazo. Un brazo que gira y se desplaza hacia arriba y abajo sobre un eje central, pero no puede mover la «mano» horizontalmente, sólo puede maniobrar en un espacio cilíndrico cuyo centro es el eje. Este tipo de *robot* se

denomina *robot* cilíndrico, y su entorno de trabajo es bastante reducido puesto que al no poder desplazarse en ninguna dirección, sólo podrá trabajar con aquellos objetos situados dentro del área formada por el cilindro en que se envuelve.

El número de ejes de movimiento del brazo determina la flexibilidad del movimiento dentro de su envolvente. El tipo de brazo mecánico utilizado en las cadenas de montaje, que aprieta las tuercas, puede tener distintos puntos de rotación, como por ejemplo en la base, la muñeca y el hombro, así como la posibilidad de abrir, cerrar y girar la pinza. Esta maniobrabilidad le permite alcanzar lugares difíciles, como por ejemplo, debajo de la guantera del coche.

Generalmente, cuantos más grados de libertad de movimientos tenga el brazo mecánico, mejor será para realizar las tareas más diversas. Sin embargo, será más caro y complicado de manejar.

El futuro inmediato de los robots se encuentra en la industria y fábricas, como elementos indispensables en las cadenas de montaje.

Los modernos *robots* de una fábrica realizan labores tediosas, repitiendo un número limitado de secuencias de movimientos una y otra vez. Los problemas que surgen al programador de *robots* es darle las instrucciones primero y luego comprobar que funcionan en la secuencia correcta en conjunción con otra maquinaria (como cadenas sin fin, otros *robots*, etc.).

Existen diversos métodos de programar un *robot*. Uno de esos métodos, por extraño que parezca, es el aprendizaje, por el que se va indicando al *robot* paso a paso la secuencia de movimientos a realizar. Esto resulta muy útil en tareas que requieren un alto grado de especialización, como pintura de carrocería. El *robot*, en el modo de aprendizaje, es desplazado a lo largo de la secuencia de movimientos que tiene que realizar, mientras el ordenador que lo controla, toma nota de las muestras de los valores de los potenciómetros situados en los ejes. Esto a su vez, genera numerosos valores que se almacenarán para ser utilizados en su momento.

Otro método es que el operador, equipado con un teclado especial, controle los motores del brazo mecánico, situándolo en la posición que estará cuando desarrolle la tarea y almacenando en el ordenador los valores de dichos desplazamientos. Como son pocos los valores indicados a lo largo de este proceso, el ordenador almacenará poca información. Sin embargo, para repetir los movimientos, el ordenador ha de calcular la trayectoria del brazo entre dos puntos, acelerando y frenando para llegar a su posición de trabajo. Resulta obvio, que cuantos más ejes tenga el brazo, más complicados serán los cálculos.

Una forma no tan común, pero que se está imponiendo poco a poco, son los lenguajes de programación especialmente preparados para tal fin. Esto permite al programador, especificar las posiciones y acciones, bucles de secuencia y acciones condicionales a seguir en el proceso. La gran dificultad que tienen estos lenguajes es que están orientados especialmente a uno o varios modelos y no se pueden emplear con la mayoría de brazos existentes en el mercado.



Foto 2. El cartucho esconde un lenguaje de programación específico para controlar este robot; el ROGO.

Los brazos mecánicos no son los únicos *robots*. El otro grupo principal lo componen los *robots* móviles. El *robot* móvil comercial es, con frecuencia, una máquina en una plataforma motorizada simple, que puede seguir unas líneas situadas a lo largo del suelo de una fábrica mediante un sensor óptico o rutas marcadas por cables enterrados cuyo campo electromagnético puede ser detectado. Tales plataformas se utilizan para transportar materiales, como la carrocería de un coche o partes de una máquina, de un lugar a otro normalmente en beneficio de los *robots* estacionarios.

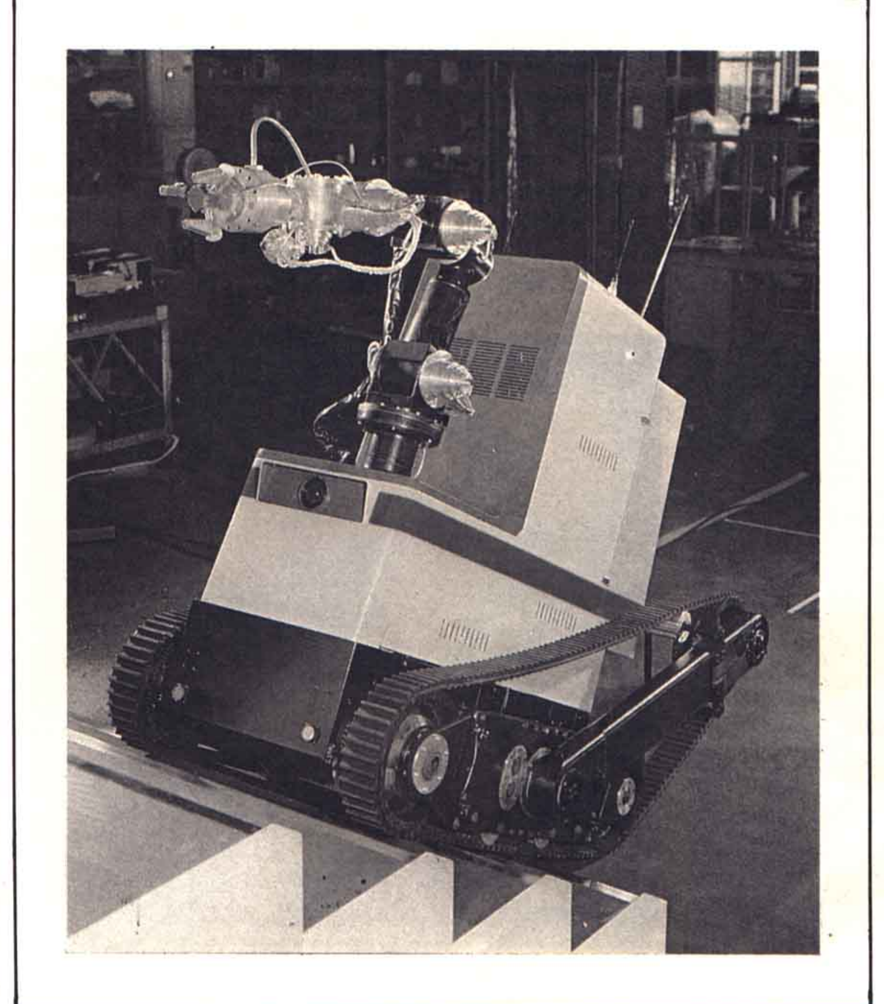
De todos los *robots* móviles existentes, el más conocido es la tortuga. Esta se controla mediante un ordenador personal, donde las señales se envían a través de un interface y un cable. La tortuga viene equipada con plumillas, que pueden ser bajadas para dibujar la ruta, con paragolpes sensibles que detectan cuando se tropieza con algo y con sensores ópticos para seguir el rastro de una línea.

Normalmente, para programar un *robot* móvil para que ejecute movimientos simples, se utiliza un lenguaje de programación de alto nivel como es el *LOGO*. Este tiene instrucciones que desplazan la tortuga hacia arriba, abajo, derecha o izquierda. Los argumentos de las instrucciones indican el número de unidades de distancia a mover o los grados a girar.

LOGO es un lenguaje de programación completo y potente, su aplicación a la robótica deriva de la utilización de la tortuga. Su facilidad para definir procedimientos, su recursividad y su proceso de listas, lo convierten en el lenguaje ideal para las aplicaciones avanzadas de *robótica,* aunque este lenguaje en principio se pensó como ayuda para la enseñanza de las matemáticas.

Controlar un *robot,* es como manejar una impresora. Los datos y caracteres de control se han de enviar por un canal de salida, mientras que por un canal de entrada se van mandando al ordenador aquellos valores que el brazo mecánico va muestreando.

El primer sentido que se ha dotado a un *robot* ha sido el tacto. Aunque de una manera no muy ortodoxa, el tacto se implementó con los detectores que indican cuando la tortuga golpea un objeto. Esto se puede conseguir utilizando unos detectores situados en el paragolpes de la tortuga, generando con su señal unas interrupciones o siendo detectado mediante *software* para iniciar rutinas que eviten al *robot* completar su tarea.



Una manera más sofisticada de proporcionar el tacto es madiante el uso de pinzas. Estas se sitúan al final del brazo mecánico, de manera que controlan su presión al coger un objeto. En algunas ocasiones, habrá visto un inmenso brazo mecánico cogiendo entres sus grandes pinzas un huevo sin romperlo, algo totalmente imposible si no se tiene el sentido del tacto.

Una tarea muy común del brazo mecánico es el montaje de máquinas ensamblando piezas. El detector de presión, colocado en la muñeca del *robot* o en la plataforma de trabajo, indica al *software* de control, la manera en que se está realizando la tarea con respecto al trabajo final. Una presión excesiva, hará que el *robot* falle al insertar una pieza en su sitio, o no cumpla con su cometido, esto es suficiente para hacer ejecutar una rutina que permita entrar en funcionamiento todos los elementos del *robot* para cumplir el objetivo.

Si nos movemos a un nivel más alto, en cuanto a sofisticación, nos encontraremos aparatos con sensores de proximidad, que pueden indicar al *software* de control la distancia existente entre la pinza y el objeto. Normalmente son dispositivos de rayos láser, infra-rojos o ultrasónicos, que transmiten una onda de radiación y luego miden la energía reflejada. Actualmente existen sensores químicos que, por ejemplo, indican la concentración de un compuesto particular

en un líquido o gas, y que proporciona al *robot* el sentido del gusto y olfato. Esto junto con contadores geiger y detectores de presión, están siendo incorporados en *robots* diseñados para tareas muy específicas y delicadas dentro de la industria, como es el manejo de materiales radiactivos y trabajos de mantenimiento en el mar. Sin embargo, el sentido más útil de todos, la vista, todavía está sin conseguir.

El proceso visual en los sistemas de *robot*, es principalmente la posibilidad de detectar las esquinas de la imagen y construir el perfil del posible objeto. La imagen en sí, está codificada como los datos obtenidos de la digitalización de una imagen de vídeo, que almacenará el ordenador antes de realizar cualquier cálculo. Al analizar esta imagen, normalmente se empieza buscando el perfil y el contorno del objeto.

En esta fase del análisis, la necesidad obliga a utilizar algoritmos muy rápidos que permitan encontrar y mostrar los bordes del objeto en la imagen. El resultado es como un dibujo realizado con líneas, de la imagen inicial y por este motivo se le conoce como el borrador en 2-D. Algunos sistemas obtienen información sobre la sombra del objeto en la imagen

Las tareas monótonas y tediosas que, normalmente viene realizando el hombre, son llevadas a cabo por estos serers fríos e inanimados.

y añaden esto al análisis para generar un borrador en dos dimensiones y media; esto es, un dibujo con información sobre las tres dimensiones del objeto pero que todavía no está representado en 3-D, debido a que se desconoce la cara oculta del objeto.

Para la mayoría de las aplicaciones industriales, tales como identificación de piezas es una cadena sin fin, y el control de orientación de las piezas, este nivel de análisis es muy adecuado. Los objetos buscados tienen su propia representación en la memoria del ordenador y un algoritmo de diseño es empleado para igualar el borrador de 2-D ó 2.5-D a la información almacenada, efectuando dos o tres rotaciones en 3-D de la representación. Para ayudar a obtener el contorno del objeto, se han diseñado sistemas especiales de iluminación que permiten resaltar el perímetro del objeto claramente. También existen sistemas de visión binocular, donde la disparidad de las imágenes obtenidas entre uno y otro foco de visión indican la profundidad del objeto, tamaño, forma y posición.

Esta clase de proceso visual requiere efectuar una cantidad importante de operaciones y no serán de fiar hasta que existan procesadores en paralelo.



Foto 3. Detalle del funcionamiento del imán.



Foto 4. Una pinza, un pala y un imán, le convierten en el compañero ideal para iniciarse en la robótica.

La gran didficultad acerca de los sentidos de los *robots* no es proporcionar el sensor en sí, sino interpretar los datos què genere dicho sensor. Un codificador óptico generará una serie de bits, pero el ordenador deberá interpretar estos como rotaciones de ruedas o como movimiento de los ejes. Es más, la salida de varios codificadores ópticos, sensores de presión y potenciómetros, han de estar integrados si han de proporcionar al *robot* la sensación de movimiento de sus piezas (esto es esencial para aquellos *robots* que han de mantenerse en equilibrio).

En realidad, hemos llegado a un punto donde el progreso en *robótica* ha de esperar al progreso en Inteligencia Artificial. Todo depende del futuro y, sobre todo, de las aplicaciones que se consigan en este campo.

Una de las metas más importante de la Inteligencia Artificial es que los ordenadores aprendan de sus errores, pudiendo cambiar su actuación, si así lo hacen las circunstancias.

Actualmente, los usuarios de MSX tienen la posibilidad de profundizar más en el tema de la *robótica* gracias al *Robot Arm,* de Spectravídeo. Este brazo se puede utilizar conectado al ordenador o independiente de él.

A corto y largo plazo, se emplearán robots en las plantas nucleares y laboratorios químicos.

Para conectarlo al ordenador, se emplea un cartucho ROM, donde se almacena un lenguaje de programación especialmente preparado para usar el *robot,* y dos cables. Estos van a la base del *robot,* donde hay dos conectores de joystick.

La programación del robot se efectúa en *ROGO,* un lenguaje derivado del *LOGO* y cuyo aprendizaje, al igual que el resto de los lenguajes, lleva su tiempo. Sin embargo, gracias a la similitud entre los dos lenguajes y la posibilidad de controlar los movimientos del *robot* mediante la pantalla del TV (existe una opción del *ROGO* en la que se van mostrando los movimientos del *robot* a unas determinadas instrucciones). Estas son similares a las que se utilizan en *LOGO,* siendo también un lenguaje recursivo y procedimental (ver artículo de «LOGO, un lenguaje educativo» publicado en MSX Magazine, núm. 11).

Si el usuario prefiere utilizar el *robot* sin el ordenador, tendrá que estar provisto de dos joysticks. De cualquier manera como una imagen vale más que mil palabras, el artículo está ilustrado con las diversas posibilidades del *robot.* Utilizando la pinza, el imán o la pala. Estas son sólo algunas de sus aplicaciones.



Foto 5. Detalle del funcionamiento de la pinza.

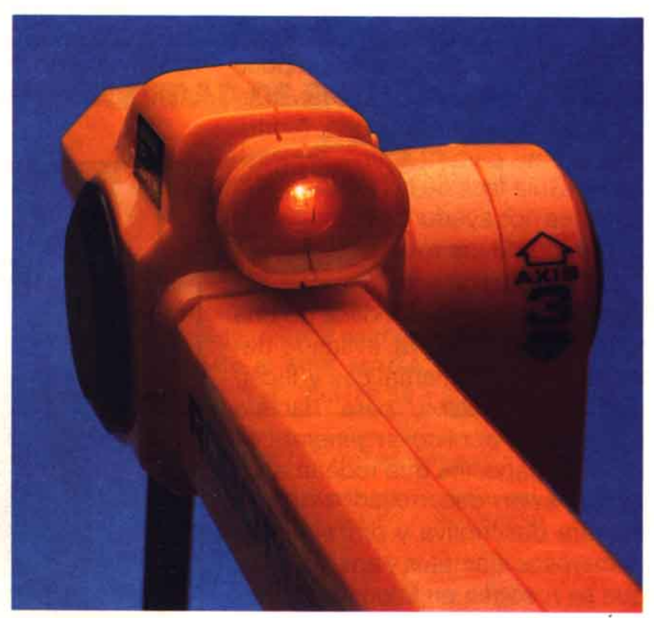
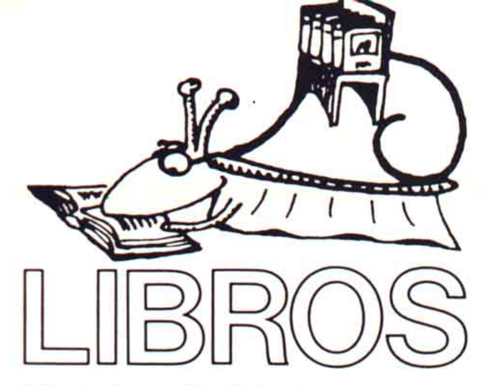


Foto 6. El pequeño robot dispone de un foco para ayudar en las tareas nocturnas.



LIBROS

Libro: Guía Fácil de la Robótica
Autores: J. M. Angulo, José Nó
Editorial: Paraninfo, S.A.
Páginas:

Agazapado en la oscuridad de tu dormitorio, con un sudor frío corriendo por tu frente, tratas de dominar tus nervios y encontrar una salida a tu situación. Lo que tantas veces habías comentado en broma con tus amigos, se está haciendo realidad. ¡Tu robot de plástico y tu ordenador se han aliado contra tí!

Y de repente, te despiertas y te das cuenta de que todo ha sido una pesadilla... ¡vaya susto! Y te empiezas a reir, diciéndote a ti mismo: «Pero que tonto soy; ¿cómo me va a atacar un robot de juguete?»

De todos modos, en el subconsciente colectivo de la humanidad, sigue existiendo un oculto temor hacia esos amigables seres que son los robots. Posiblemente, si todos los conociéramos tan sólo un poco mejor, todas estas pesadillas y suspicacias pasarían al olvido. Afortunadamente para nosotros, aquí llega la «Guia fácil de la Robótica», un libro que nos ayudará a comprender el mundo de los robots.

La finalidad de este libro es introducir el tema de la robótica de una forma clara y sencilla, evitando demostraciones matemáticas y tecnicismos complejos, para hacerlo asequible al público en general. Los distintos aspectos que rodean a los *robots* están desarrollados de una manera descriptiva y práctica, con numerosos ejemplos y aplicaciones que se recogen en fotografías y esquemas, y que intentan desmitificar al *robot* y facilitar tanto su conocimiento como su empleo.

Lo primero que encontramos en sus páginas es un comentario acerca de los orígenes de los *robots* y sus fundamentos, su estructura y las circunstancias que dieron lugar a su nacimiento. Poco a poco se nos van explicando los diversos elementos que intervienen en el control de los movimientos del *robot,* así como la programación y gobierno, el papel que juegan los sensores enviando información del mundo exterior a la unidad encargada del control de los movimientos, y la unidad fundamental de la Inteligencia Artificial en el control de estos aparatos.

También encontraremos una exposición de las diferentes áreas en las que se está usando estos dispositivos, los requisitos previos que se deben satisfacer para la instalación de *robots* en las factorías, la metodología que se debe seguir para su implantación y el estado actual del mercado de *robots* en el mundo.

Al final de cada capítulo se incluye una bibliografía con libros que nos ayudarán a ampliar nuestros conocimientos en estos temas.



En definitiva, se trata de un libro de interés general, muy adecuado para introducirse en el mundo de los robots con unas perspectivas y un punto de vista amplios.

Libro: Códigos y Claves Secretas. Criptografía en BASIC
Autor: Gareth Greenwood
Editorial: Anaya
Páginas:

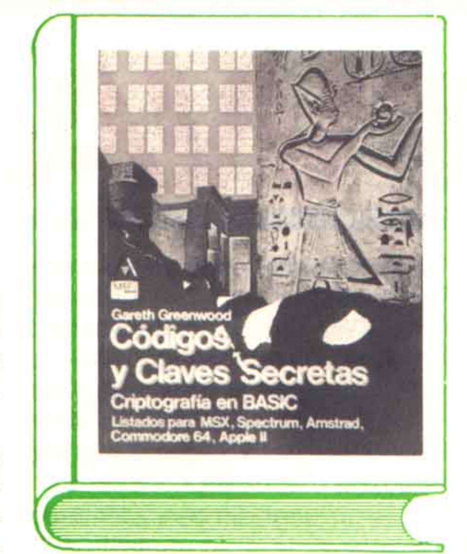
La escritura secreta es conocida desde tiempos inmemoriables. En el antiguo Egipto había dos tipos de escritura: la gente normal usaba la llamada escritura *demótica,* pero posiblemente para mantener en secreto los rituales religiosos, los sacerdotes usaban una forma de escritura totalmente diferente, llamada escritura *hierética.* Muchos de los líderes militares de la Antigüedad utilizaron alguna forma de escritura secreta. Los espartanos usaban un sistema llamado *escítala,* que mezclaba las letras de un mensaje para que el enemigo no pudiera leerlo si eran capturados. Julio César usaba una simple escritura secreta que cambiaba las letras del escrito de forma que quedaba aparentemente ininteligible. A lo largo de la historia siempre ha habido gente con algún motivo para guardar un secreto de ojos ajenos. Casi siempre usaron códigos y escrituras secretas para conseguir sus objetivos.

El estudio de las escrituras secretas es una ciencia por derecho propio. Se llama *criptología.* Hay dos ramas principales en la criptología, llamadas *criptografía* y *criptoanálisis.* Las palabras se derivan del griego antiguo (*Kryptos* significa secreto o escondido) e incluyen todos los métodos de codificación y cifrado que se pueden usar para hacer se-

creto un texto. El criptoanálisis intenta analizar qué es lo que se ha ocultado, es decir, descifrar las claves y protecciones de forma que se pueda leer el secreto del enemigo.

La criptología y el criptoanálisis han jugado papeles muy importantes en muchos de los grandes acontecimientos de la historia, aunque no siempre se le ha dado publicidad. Esto es particularmente cierto en materia militar y diplomática. Se han ganado y perdido batallas por el desciframiento de un mensaje. La escritura secreta ha guardado algunos de los mayores secretos políticos que el mundo ha conocido (o incluso que no ha llegado todavía a conocer).

Las grandes potencias del mundo continúan usando la criptografía, pero ésta ya no es del uso exclusivo de Gobiernos y Ejércitos. Sí —lo habrás adivinado—, los ordenadores son el motivo. Los usuarios de los ordenadores se han dado cuenta de que necesitan proteger los datos que almacenan en los bancos de datos o que transmiten a lo largo de las líneas de telecomunicación. ¿Cómo pueden protegerlos? Poniéndolos en clave. El *microchip* ha dado repentinamente una nueva dimensión a la criptología, y ha hecho de ella una cosa natural.



Este libro supone una introducción a la criptología para los usuarios de ordenadores personales de bajo precio.

Se muestran una serie de cifrados y se acompañan los programas en BASIC, que permiten al ordenador codificar y decodificar estos mensajes. Otros programas permiten separar en distintas partes mensajes desconocidos para poder leerlos. Este libro está escrito para todos aquellos a los que les gusten los rompecabezas, ya que decodificar un mensaje no es más que un gran rompecabezas.

Los listados se ofrecen en versión para AMSTRAD, MSX, Spectrum, Commodore 64 y Apple II.

SOFTWARE

Programa: Hiper Sports 3
Tipo: Juego
Distribuidor: Konami
Formato: Cassette

Konami sigue en el mercado, cada día con mayores alicientes y mejores programas, aunque siga un poco la trayectoria sucesiva de juegos del mismo tema. *Hiper Sports,* es la tercera parte de un compendio de deportes en los que se intenta dar gusto a todos y llevar al mundo del software la Olimpiada más espectacular y divertida.

Mirando un poco hacia atrás, esta trilogía comenzó con deportes usuales, que tenían participación en todos los juegos olímpicos, como podían ser el salto de altura, carreras de velocidad e incluso la halterofilia. Por ello, basándose siempre en el camino andado, se introducen ahora nuevos deportes de mayor embergadura e incluso más populares, como el ciclismo.

En este juego, encontraremos diversión y satisfacción, sobre todo a la hora de haber superado una marca y poner nuestro récord personal como baremo de la puntuación.

Es un juego sin particularidades a la hora de jugar, puesto que el control realizado tanto por joystick o cursores es fácil y no se pierde precisión en los movimientos empleando cualquiera de los dos elementos.

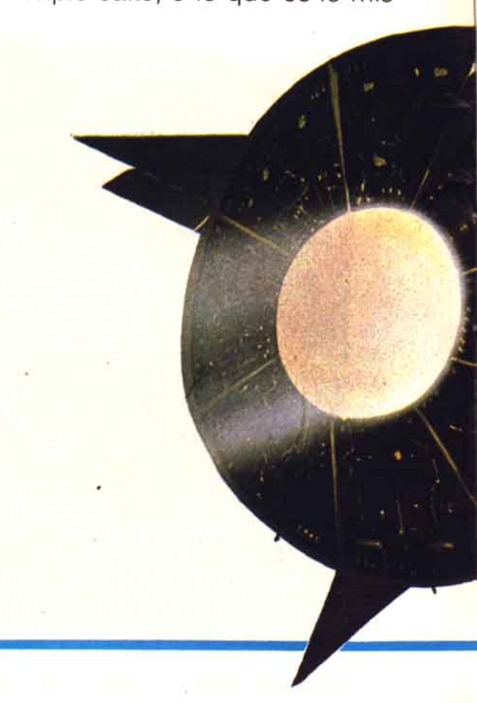
Analizando la parte intrínseca del programa o aquellas características que lo forman, debemos hacer mención a la musicalidad que acompaña a este tipo de programas, ya que decir sonido, no será justo. Las sintonías de comienzo y fin del programa, nos demostrará la singularidad de todos los programas de la serie. El sonido de los aplausos es otro de los elementos que configuran el programa, acompañado en todo momento por la claridad de imagen y el colorido excepcional que acompaña a todas las pruebas, aunque el deporte del ciclismo, prueba estelar de este programa, no tenga el cáriz idóneo, en cuanto a color y espectación que hubieramos deseado.

Ciclismo es la primera prueba que se nos presenta. Una salvedad: todas las pruebas son eliminatorias, así que en cada una de ellas debes poner todo tu esfuerzo para poder pasar a la siguiente. Son 2.000 m de dura puja por el primer puesto, además de tener que salvar obstáculos y numerosos contrincantes que intentarán que te salgas de la pista.

No es un circuito de un estadio de deportes, sino que la prueba transcurre por una carretera al aire libre para poder oxigenar bien los pulmones. Cuando aparezca una pistola en la pantalla y la señal de salida, pulse lo más rápidamente posible el botón de su joystick y acuérdese de esquivar los obstáculos y en la pantalla aparecerá una flecha que le indicará hacia qué lado debe moverse. No existe en esta primera competición prueba de puntos, pero deberás realizar la carrera en un tiempo menor a tres minutos.

Triple salto, o lo que es lo mis-





mo: ¡hop, step and jump! avance, pise y salte, una perfecta combinación para conseguir la mayor distancia en el salto. Esta es la segunda prueba, y no menos difícil.

La tercera prueba es toda una innovación, *CURLING* o deslizamiento. Este es un juego de precisión en el que debes hacer deslizar una piedra suavemente por el hielo, hacia al blanco. Tus dos compañeros de equipo, corren delante de ti, barriendo furiosamente el hielo, para que no se desvíe y llegue al centro consiguiendo así la mayor puntuación. Primero debes elegir la temperatura del hielo, ya que si está a una temperatura muy alta tendrás que empujar con más fuerza para que se deslice. Ten mucho cuidado con esta insignificancia.

La última prueba es el salto de pértiga. Corre lo más rápidamente posible y acércate a la barra. Esta prueba, aunque parece sencilla es la más dura, ya que debes ser muy exacto tanto en la fuerza que desarrolles como en la inclinación que tomes. Y ahora sólo queda oír la músicaa de los campeones de Konami.

Puntuación:
Presentación: 8
Claridad: 7
Rapidez: 7
Adicción: 9

Programa: Pitfall II
Tipo: Juego
Distribuidor: Proein, S.A.
Formato: Cassette

Harry, el héroe de nuestra aventura, se encuentra en una peligrosa caverna en la que sin darse cuenta se ha caído; numerosos son los peligros que encontrará a su paso, ya que además de defenderse y luchar con los más horribles monstruos del fondo de la tierra, tendrá que soportar el cansancio y la fatiga ante la falta de oxígeno según va descenciendo por la caverna. Es un juego rápido, basta un simple descuido, para ser derribado por un monstruo y perder todos los puntos que habíamos conseguido. El porcentaje de dejarnos capturar es muy bajo, ya que todos los peligros que encontramos a nuestro paso se mueven muy despacio y muy calculadamente. Esto no quiere decir que no le prestemos atención por su facilidad, sino que antes de saltar un monstruo o entorpecer su camino calculemos todos los detalles.

Señalaremos que una de las principales características del juego es que la aventura se subdivide a su vez en fases, no en fases de distinto contenido, sino en que una vez recorrido una parte del juego, será una etapa realizada que ya no podremos perder. Existen unas cruces en el suelo que son las que nos indican las etapas



SOFTWARE

de la aventura. Cada vez que lleguemos a una de ellas habremos «dado un paso adelante». Cada vez que nos elimene un objeto volveremos a la cruz más cercana que hayamos dejado detrás.

El escenario del juego, como hemos dicho antes, es una caverna, pero es una caverna especial, porque *Rhonda,* nuestra compañera de juego y su gato *Quickclaw,* han desaparecido dentro de ella. Nuestra misión es encontrarlos y junto a ellos el diamante *Raj,* el cual nos concederá todo el poder para poder salir con vida de esta insospechable aventura.

Hacer un croquis del jugo antes de comenzar a jugar, os ayudará mucho ver la demostración para ver la subdivisión en niveles de la caverna. Para aquellos más lanzados, podéis experimentar vosotros mismos, ya que en este juego no hay vidas y sí... mucho tiempo.

No bajéis nunca ningún nivel sin haber cogido el diamante que se encuentra allí o por lo menos haberle echado un vistazo, pues si no conseguís todos los diamantes no pasareís de pantalla y entonces será interminable.

Los monstruos que aparecen (lagartos, murciélagos) siempre desarrollan un movimiento uniforme, calcular y saltarlos, pero cuidado: para bajar de nivel existen escaleras en unos agujeros que se comunican entre niveles. Agarraros bien o caeréis infinitamente hasta chocar con un monstruo, o lo que es peor, iréis a parar al gran lago subterráneo del que os será difícil salir. Pero la aventura continúa, numerosos serán los alicientes que encontraréis, así que adelante, pues una característica de este juego es que la aventura no termina nunca hasta que tú no decidas apagar tu ordenador.

Los gráficos son aceptables en cuanto a su realización, pues *Harry* es todo un aventurero; la amplitud de visión de los distintos niveles también nos ayuda mucho, es un perfecto croquis que no nos depara sorpresas, aunque hay que ser muy rápido. Sólo debemos hacer una salvedad: el color, ya que para nuestro gusto es poco significativo, pero es un detalle ítimo.

Puntuación:
Presentación: 8
Claridad: 8
Rapidez: 6
Adicción: 8

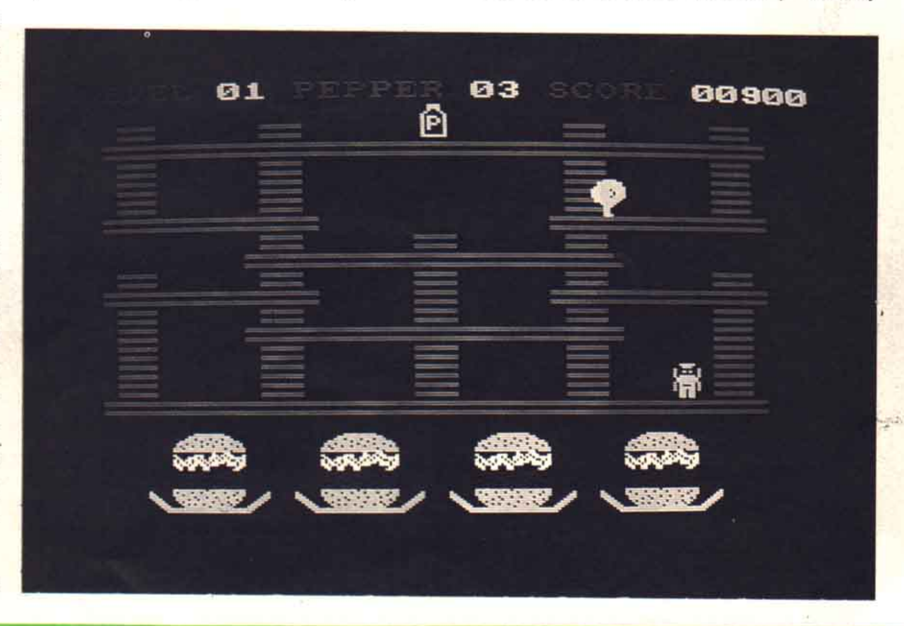
Programa: MC Attack
Distribuidor: DATAMARKET
Tipo: Juego
Formato: Cassette

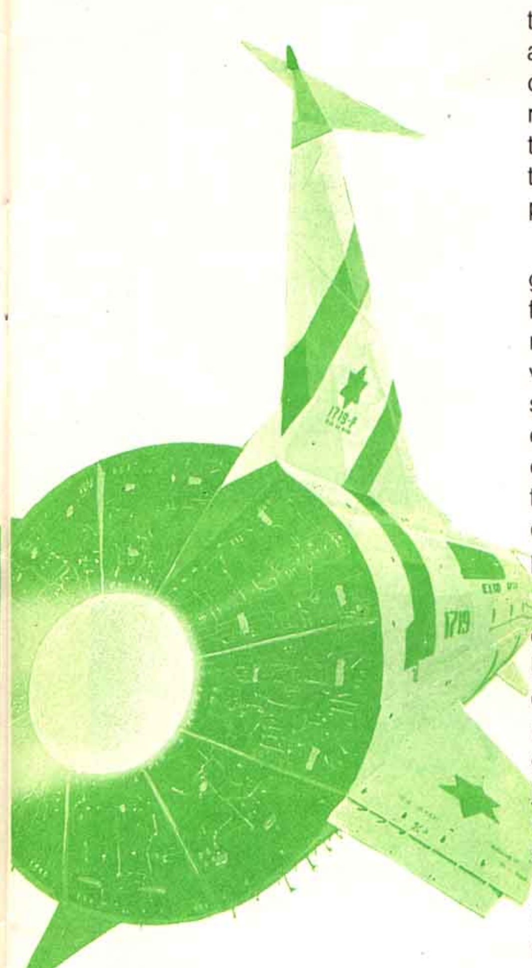
Nos encontramos frente a un juego que por sus características no podemos englobar en ningún bloque o apartado. Es un juego con vida propia e innovador.

No es que el tema que trate sea fascinante y deseado por los jugadores de estos juegos para ordenador, sino que una serie de elementos como música, color, gráficos y diversión le hacen diferente a los demásd y ocupa a nuestro entendimiento un lugar entre los más divertidos y sencillos juegos.

El propio nombre del juego nos despista un tanto de las características, ya que hubieramos deseado fuera Freddy, pues es él el protagonista. La acción se desarrolla en una hamburguesería, no parecía que el tema fuera éste, pero cuando juegues la primera partida verás que es un juego entretenido y sobre todo diferente, aunque hoy en día, las posibilidades estén saturadas.

Freddy es el mejor cocinero de Hamburguesas, pero algunos alimentos se han revelado contra él y no dejan servir sus hamburguesas con la rapidez que Freddy querría. Un huevo frito y una salchicha serán nuestros enemigos inminentes, ya que todos los componentes de nuestra hamburguesa se encuentran en diferentes niveles, y Freddy





debe conseguir que vayan bajando y caígan todos colocados en sus platos y listos para servir.

La salchicha y el huevo frito irán soltando grasa a fin de estorbarnos en el camino y ocasionarnos una desgracia, pero Freddy que debe ser más hábil que todos ellos, como buen cocinero sabe darles el tiempo justo de preparación a las hamburguesas y saldrán listas para servir cuando sea necesario.

Infinidad son los elementos que podríamos resaltar de este sencillo juego, comenzaremos por las generalidades que presenta para terminar con las características que le hacen singular.

Todos los juegos en los que la meta sea sumar puntos y más puntos carecen en cierto modo de un aliciente que nos satisfaga a la hora de concluirlo, es por ello que desarrollar el cometido de Freddy, no nos transporte a una maravillosa aventura, pero es que no todos los juegos pueden ser iguales, y éste no lo es.

La forma de presentación de los gráficos, en su totalidad utilizan la forma tradicional de presentar varios pisos por los que debemos movernos con la mayor rapidez para no ser alcanzados, y sólo contaremos con un único elemento para defendernos, la pimienta que hiere mortalmente si la dosis es muy elevada o bien paraliza por unos segundos para que podamos escapar.

¿Qué sensación le produce ver corrieno a un huevo frito detrás de Ud.?, desde luego puede ser muy divertida. Todos los sprites que aparecen y su animación o movimiento, es sin duda uno de los alicientes del juego, nuestro huevo frito resbala cada vez que tiene que correr, o la salchicha se dobla para subir las escaleras, pero todo está unido a un excelente color y realizado con gran detalle. Se han preocupado de hacer buen programa.

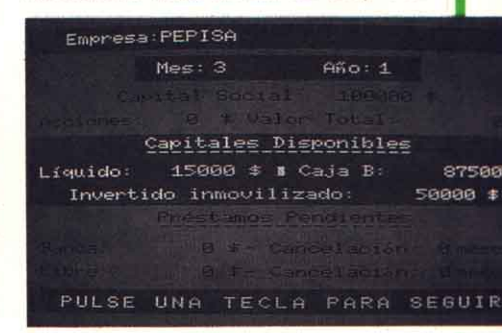
La nota que debemos resaltar es el sonido, han creado un efecto de eco que proporciona al juego una característica técnica que hasta ahora pocos habían conseguido, esperemos que pueda disfrutar de él.

No podemos mencionar más detalles que le sirvan para divertirse más, ahora bien, se lo recomendamos, es un juego interesante, sencillo y entretenido.

Puntuación:
Presentación: 8
Claridad: 8
Rapidez: 6
Adicción: 7

Programa: Risky Holding
Distribuidor: Dimensión New
Tipo: Juego
Formato: Cassette

Personas frías y calculadoras serán los únicos que puedan jugar. Es un juego para los amantes del riesgo y la aventura dentro del complejo mundo de las finanzas, puesto que no sólo se trata de ser un empresa-

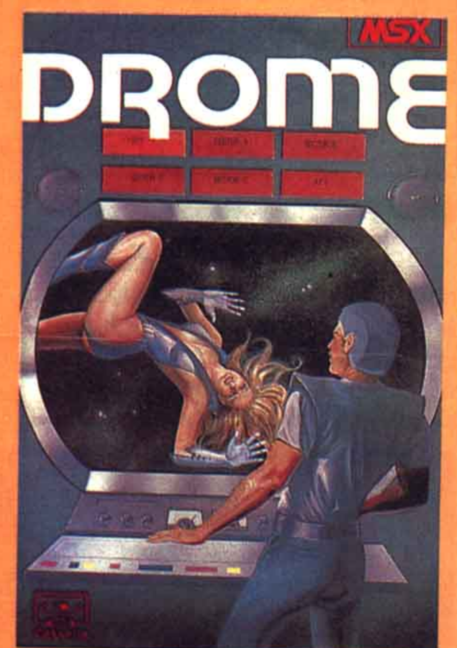


rio, sino de crear un imperio basado en el esfuerzo empresarial y utilizando las más sofisticadas estrategias, haciendo bueno la frase «el fin justifica los medios».

El Risky Holding, juega el papel del empresario prototipo, que no escatima medios para realizar operaciones arriesgadas y conseguir hacer de sus pequeños ahorros un Holding. Riesgo sin duda se presentará al ver que no sólo cuenta con operaciones legales, sino que a su vez con una parte de la economía que todos sabemos y pocos conocemos; la economía sumergida.

Este juego es una simulación, muy buena, de lo que supone mantener una empresa en lo más alto del ranking y de los medios que cuenta para ello, aunque deje de lado algunos aspectos de la economía, como pueden ser las relaciones laborales, compras, ventas, etc.

el mejor softwa





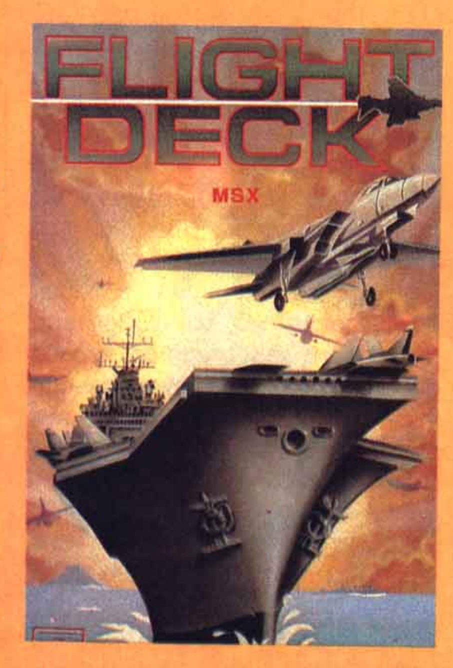
DROME

Entretanto en DROME, un Super-ordenador, debes encontrar y eliminar los sofisticados sistemas de defensa y supervivencia.

Has de elegir uno de los cuatro sectores que constituyen los mecanismos de defensa de esta terrorífica máquina.

Un atractivo juego de acción, donde se pone a prueba la capacidad de la máquina y del jugador.

Precio de venta 2.600 ptas. (IVA incluido)



FLIGHT DECK

Sienta la emoción del golfo de Sidra en casa.

FLIGHT DECK es un juego de estrategia y habilidiad en el que tendrás que desmantelar las bases enemigas.

Al mando de un portaaviones donde dispones de 10 unidades de combate... y poco tiempo.

Precio de venta 2.600 ptas. (IVA incluido)





MC-ATTACK

Ayuda a Fredy, el Rey de la Hamburguesa a preparar el suculento manjar que hace las delicias de los comensales.

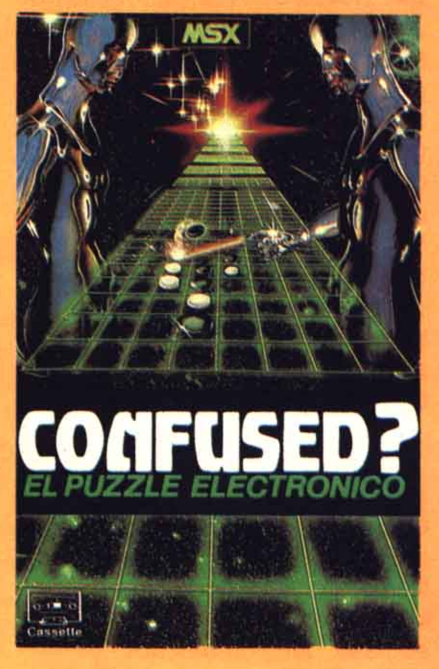
Ten cuidado con las salchichas grasientas y los huevos escurridizos que intentarán arruinar tu exquisito plato.

Defínete con la pimienta y procura hacer el mejor número de hamburguesas posible.

... Buen provecho.

Precio de venta 1.900 ptas. (IVA incluido)

re para MSX



CONFUSED?

Es el puzzle electrónico.

El objeto del juego es resolver 10 puzzles con distinto número de piezas, a elegir, pero todas... MOVIENDOSE.

Pon a prueba tu inteligencia y capacidad de deducción para solucionar algunos de estos entretenidos rompecabezas.

Precio de venta 2.600 ptas. (**IVA incluido**)

NORTH SEA HELICOPTER

Una explosión en una plataforma en el mar del Norte arroja a los hombres a un destino incierto...

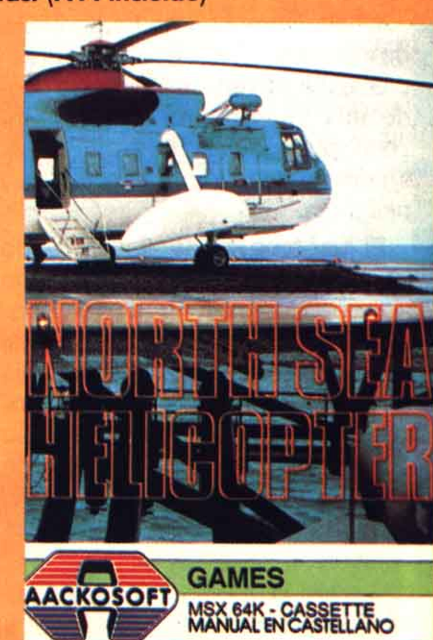
Empieza una carrera contrareloj para salvarles de su fatal situación.

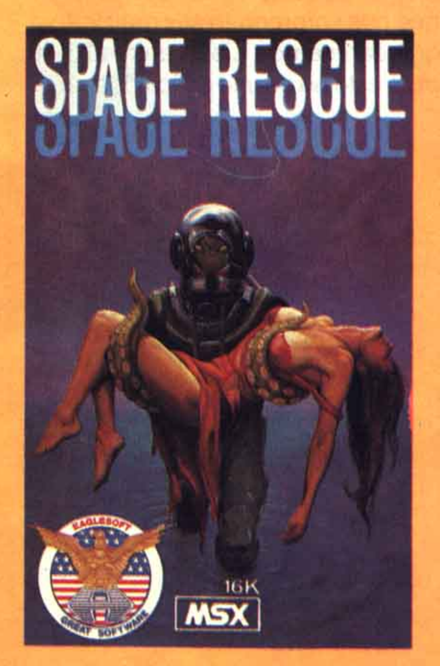
Tienes que convertirte en un piloto experimentado para mantener el control del helicóptero... El tiempo empeora.

¿Crees que cumplirás la misión?

Precio de venta 2.600 ptas. (**IVA incluido**)







SPACE RESCUE

Desciende de tu nave nodriza a la superficie lunar e intenta rescatar a los hombres extraviados.

Ojo con los meteoritos que te destruirán cuando intentes regresar.

Disponer de nuevas plataformas para culminar con éxito la misión.

Desgraciadamente tu nave nodriza está bloqueada por unidades de combate enemigas... Intenta destruirlas.

Precio de venta 2.600 ptas. (**IVA incluido**)

Solicita tu programa favorito en tu tienda especializada habitual o recorta y envia este cupón a INFODIS, S. A., c/Bravo Murillo, 377. 5.º A - 28020 MADRID.

NOMBRE ____
DIRECCION ____
POBLACION ____ PROVINCIA ____ C. P. ____
Forma de pago: CHEQUE ☐ CONTRAREEMBOLSO ☐
TARJETA DE CREDITO: American Express ☐ VISA ☐ INTERBANK ☐
Número de mi tarjeta ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
CADUCIDAD ____
Firma ____

TITULO	CANTIDAD	TITULO	CANTIDAD
DROME FLIGH DECK MC-ATTACK		CONFUSED? NORTH SEA HELICOPTER SPACE RESCUE	

infodis, s.a.

SOFTWARE

Pero este tipo de empresa no es a la que nos estamos refiriendo, nuestro Holding será una empresa especulativa, cuyo fin es la de apoderarse de dinero y más dinero, sin detenerse en los medios de conseguirlo.

Nos presenta la posibilidad de ser uno de los cuatro empresarios (número máximo de jugadores) que con su esfuerzo debe llegar a controlar la mayoría de las acciones del Banco Universal, junto con esto y una gran liquidez podemos llegar a dominarlo todo.

Pero no contaremos con ayuda alguna, tan sólo nuestra intuición y a veces la suerte, será necesaria para salir adelante de tan dura empresa.

Contaremos con un fondo inicial de cien mil dólares, así como un capital especial del mismo valor y desde ahí comenzaremos a realizar inversiones o incluso arriesgar nuestro dinero de la forma más osada y peligrosa.

El juego se desarrolla indicando los turnos de jugadores, para que así no se cometa ningún error. Aparecerá una tabla general del capital social, dinero líquido, caja B o caja de fondos obtenidos en operaciones ilegales, inmovilizado en curso o inversiones y créditos obtenidos en organismos oficiales (Bancos) o en el mercado libre. Esta pantalla mostrará la visión global de la empresa, pudiendo ser accedida en cualquier momento de la partida y consultarla antes de realizar cualquier operación. Ahora bien, cuando estemos realizando una operación, no podremos echarnos atrás, si no es por falta de dinero, puesto que cada una de éstas tiene un mínimo estipulado. Una vez que sea nuestro turno podremos escoger entre diferentes opciones para realizar una operación financiera; compra-venta de acciones, operaciones legales, operaciones sumergidas, etc.

Cada una de ellas se divide, a su vez, en varias opciones. Comentaremos sólo las operaciones legales. En ellas podemos invertir en Certificados de Depósito, con un 20% de beneficio a 3 meses; Bolsa de Metales, de las mismas características; Certificados de Depósito, convertibles a seis meses y Piedras Preciosas con la mayor rentabilidad de todas, hasta un 40%.

En cuanto al apartado de operaciones ilegales, no vamos a desvelar los secretos, pero los beneficios que se pueden obtener van desde el 150% hasta el 300%.

Como toda empresa, entran en juego las normas existentes en el país donde ésta esté ubicada. Así podrá encontrarse en situaciones límites como golpes de estado, nacionalizaciones, inundaciones, etc. Todo esto deberás soportarlo con tus recursos financieros.

Risky Holding, es verdaderamente una estrategia a realizar en el mundo de las finanzas. Aunque su campo sea muy limitado y el aspecto competitivo entre empresarios no se vea bien reflejado, de todas formas, es un juego y como tal es divertido, entretenido y arriesgado en todo el contenido de la palabra.

Arriésguese y recuerde, «el fin justifica los medios».

Puntuación:
Presentación: 7
Claridad: 8
Rapidez: 8
Adicción: 9

Programa: Space Rescue
Distribuidor: Datamarket
Tipo: Juego
Formato: Cassette

Un adictivo juego de acción. Es la mejor definición para este programa. Naves espaciales, intrusos de otras Galaxias, enemigos y ficción se encuentran reunidos en este juego. Por suerte, en el mercado del software tenemos infinidad de juegos galácticos para elegir, pero todos ellos siempre han presentado y presentan un carácter destructivo y hostil. Space Rescue no es diferente, e incluso diríamos que sigue las directrices marcadas por los habituales juegos de este tipo, pero ha cuidado detalles que le resaltan entre los demás.

Controlar una nave por el espacio no es tarea fácil y aún menos aterrizar. En este juego, además de tener que acertar en cada momento, tendrá que aprender a ser buen piloto y fijarse en cada momento en todos los elementos que se interponen a su paso, que son muchos.

Haciendo referencia al aspecto técnico del juego, no podemos decir que nos sorprendan sus gráficos, ya que carecen de cierta animacidad, pero esto no es una crítica destructi-



va, sino simplemente informativa de las características del juego.

En este juego encontraremos una conjunción de elementos, meteoritos, naves nodrizas, plataformas de alunizaje, etc. cuyo fin inmediato es la de rescatar los humanos que se encuentran esperando en las plataformas.

En este juego, el sonido se encuentra en la medida de lo necesario, ya que una música repetitiva o estridente nos haría perder la concentración e incluso la atención al juego. Por esto, sólo percibimos el sonido de los propulsores de la nave, los disparos y el sonido producido al estallar una plataforma o meteorito, así como la destrucción de las naves enemigas o de la propia.

El objetivo está claro, descender al planeta y rescatar a los astronautas que allí se encuentran para volver a la Tierra con todo el equipo. Evita los meteoritos y todo contacto con elementos no indentificados o que no te ofrezcan seguridad. Cuando aterrices en una de las plataformas, procura hacerlo perfectamente. Esto será complicado, porque al atravesar la lluvia de meteoritos, te encontrarás desplazado de la plataforma hacia uno u otro lado provocando, si te estrellas, la destrucción de la nave, la plataforma y los astronautas que allí se encuentren. El momento de descender será crucial, cualquier contratiempo evitará un descenso perfecto.

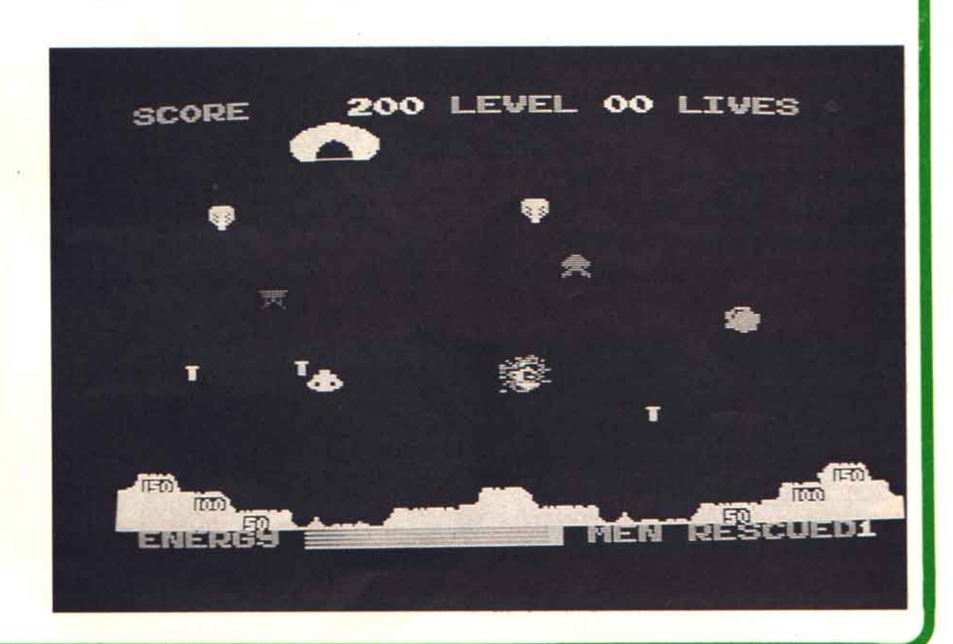
Una vez aterrizado, automáticamente subirá un astronauta, con lo que se inicia el ascenso a la nave nodriza. A partir de este momento, sólo tus rayos láser te separan del final de la misión. Destruye cuantas naves enemigas se interpongan en el camino y alcanza la nave principal con el superviviente. Aún no acaba la misión, deberás salvar a todos y cada uno de ellos antes de pasar a la siguiente fase, más difícil todavía.

Como es lógico, no se puede aterrizar a una velocidad elevada, te estrellarías de inmediato, usa el disparador o la barra de espacio para frenar el descenso de la nave. Los cursores controlan la nave, pero ¡ojo!, el cursor izquierdo desplaza la nave a la derecha y viceversa. Esto se debe a que los cursores controlan los cohetes laterales.

Si rescatas a los nueve hombres, habrás ganado un bonus, además de subir el nivel y un inagotable cañón láser.

Space Rescue es, en definitiva, un juego galáctico pero no tiene como fin destruir o salvar la independencia de la Tierra, sino una función de salvamento en lugares alejados de la superficie terrestre.

Puntuación:
Presentación: 6
Claridad: 6
Rapidez: 6
Adicción: 7





Mitsubishi ML-G1 y ML-G3

Hace unos meses, aparecieron en el mercado los primeros ordenadores MSX de la segunda generación. Primero fue *PHILIPS* con el *VG-8235* y posteriormente *SONY* con el *HB-500P.* Ahora hace su aparición en el mercado dos modelos de *MITSUBISHI* que han puesto el listón a un nivel muy alto. Los dos exponentes de esta casa son *ML-G1* y *ML-G3.* El primero es un equipo compacto, pensado especialmente como ordenador doméstico, mientras que el *ML-G3* está orientado hacia el mercado semi profesional, con el teclado separado de la unidad central y una unidad de disco (con la posibilidad de conectar una segunda unidad interna).

Mitsubishi ML-G3

Con este modelo, *MITSUBISHI* ha superado al resto de los MSX del mercado.

En efecto, este ordenador ofrece una serie de ventajas que otros ordenadores de este tipo no tienen. Cuando comenzamos a trabajar con él, tenemos la sensación de estar frente a un ordenador profesional. El tacto y su presentación así lo parecen.

El teclado que se encuentra unido a la unidad central por medio de un cable flexible, detalle que hace más cómodo su manejo. Además, está totalmente adaptado al castellano, con la facilidad de introducir, tanto la ñ como los acentos directamente, sin tener que recurrir a la tecla *SHIFT* como

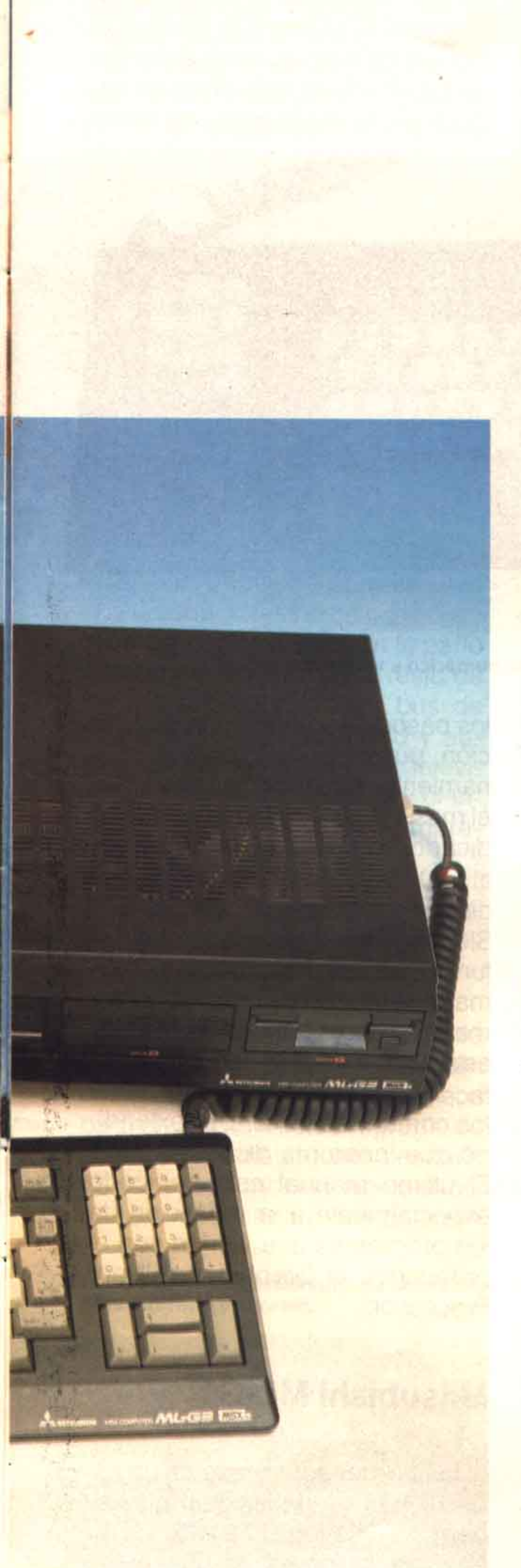




Foto 1. El conjunto que forman unidad central y teclado ML-G3 dan un aspecto profesional al ordenador.

hay que hacer en la mayoría de los ordenadores. Un teclado numérico, separado del teclado principal, le dan ese aspecto profesional del que hablábamos anteriormente. Esta cualidad facilita en gran medida la introducción de este tipo de datos.

En la parte frontal de la unidad central, encontraremos una unidad de disco de 3.5 pulgadas con una capacidad de 720K. A la izquierda de esta unidad, hay una carcasa que protege la apertura para conectar una segunda unidad. A la derecha de ésta, existe un bus de expansión para la cone-



Foto 2. Vista anterior de la unidad central con unidad de disco, carcasa para disco adicional, interruptor de conexión y reset, y bus de expansión.

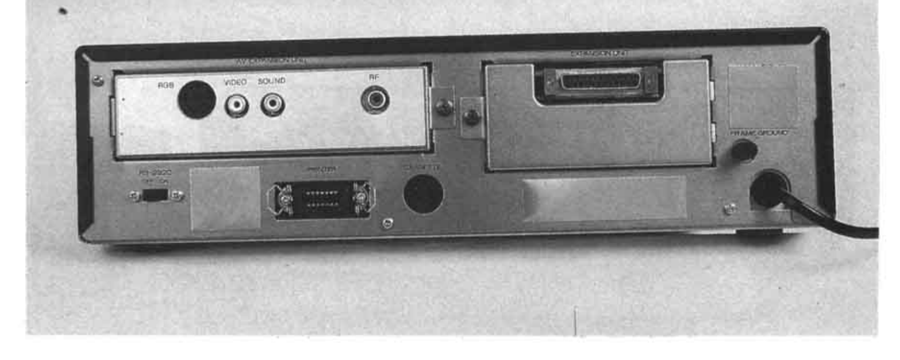


Foto 3. Vista posterior de la unidad central, donde podemos destacar todos los conectores del estándar mas el RS-232-C con su interruptor.

xión de cartuchos *ROM* e interfaces, y debajo se encuentra el interruptor del equipo y el led indicador de funcionamiento. También cuenta con una tecla *reset,* situada bajo la unidad de disco, y protegida (más si cabe) para no pulsarlo accidentalmente. En el lateral derecho de la unidad central, encontraremos los conectores de uso más frecuentes: teclado, cassette y joystick. El resto de los conectores se hayan en la parte posterior de la unidad central. En ella encontraremos los conectores estándar: centronics, bus de expansión adicional, salida RGB, salida de RF. También encontraremos un interface *RS-232C,* siendo éste uno de los pocos ordenadores que lo incorporan. Esta posibilidad permite ser conectado con otros ordenadores o periféricos, tales como modems, impresoras terminales, etc. que utilicen este interface. Este interface posee un interruptor que permite conectarlo y desconectarlo en caso de que no queramos utilizarlos. De esta manera se consigue una mayor velocidad de proceso cuando no se utilice dicho interface. El equipo se suministra con unos manuales y software de aplicación en disco, entre los que encontraremos el sistema operativo MSX-DOS, un programa de diseño gráfico y un procesador de textos.



Foto 4. Teclado independiente con teclado numérico y teclas de función.

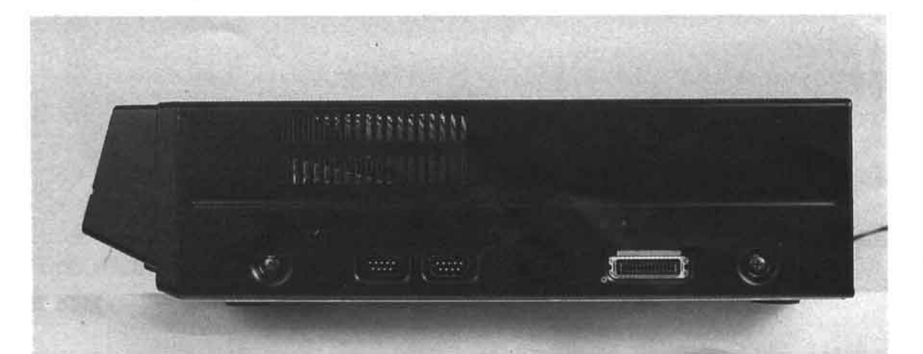
El programa «Graphic», dispone de las más diversas opciones que permiten un aprovechamiento total de las magníficas posibilidades gráficas de los ordenadores de la segunda generación.

El procesador de textos, aunque no es un programa extremadamente complejo, dispone de las opciones fundamentales en este tipo de programas.

Documentación

Con el ordenador, se incluyen nada menos que 5 manuales, cada uno dedicado a un tema en concreto. El primero de ellos es el manual de instrucciones del ordenador, donde se describen las características del aparato, así como los pasos a seguir para su instalación, puesta en marcha y funcionamiento. El segundo manual es el más extenso de todos y está dedicado, como no, al BASIC MSX y al MSX-DOS. Otro manual está dedicado íntegramente, al DISC-BASIC, explicando detalladamente el funcionamiento de todos los comandos relacionados con el sistema de disco. El siguiente manual está destinado al usuario del interface *RS-232C.* En él se describen los comandos del BASIC extendido que incorpora dicho interface. El último manual está dedicado especialmente a la utilización de los programas que vienen con el ordenador, al Graphic y al Word Processor.

Foto 5. Vista del lateral derecho con los conectores de joysticks y del teclado.



Mitsubishi ML-G1

Este ordenador presenta un aspecto menos profesional que su hermano mayor, el *ML-G3,* y a diferencia de aquél es un equipo



Foto 6. **El pequeño de la casa, con prestaciones a lo grande, se denomina ML-G1.**

compacto que incorpora en la misma carcasa el teclado y la unidad central. El *ML-G1* está diseñado como ordenador doméstico, y con el objeto de hacerlo más asequible se ha suprimido la unidad de disco y el interface *RS-232 C.*

Sin embargo, esto no le resta ni elegancia en el diseño ni posibilidades al equipo, es simplemente un ordenador descafeinado de la segunda generación.

El teclado es idéntico al modelo superior *ML-G3* (tanto la ñ y el acento se introducen directamente), la tecla *reset* se encuentra en el lateral derecho del aparato junto con los conectores para joysticks y la salida para cassette. En la parte posterior encontramos el resto de los conectores estándar; bus de expansión, salida de vídeo y audio, salida modulada para televisión, salida de vídeo RGB y el interface centronics para la impresora.

Software en ROM

El *ML-G1* incorpora en 48K de ROM el mismo programa de gráficos (Graphics) que viene en disco para el *ML-G3.* A este programa se accede inmediatamente después de poner en funcionamiento el ordenador, de manera que para entrar al BASIC, tendremos que utilizar la opción «END» de dicho programa o mantener pulsada la tecla «DEL» mientras dura el proceso de inicialización. El BASIC de este ordenador es el mismo que el *ML-G3,* y los manuales son los mismos a excepción de los dedicados al BASIC-DISC y al *RS-232C.*

Conclusión

Aunque básicamente ambos ordenadores estén derivados de un mismo modelo, cada uno está orientado a un tipo diferente de usuario. El *ML-G1* sirve perfectamente como ordenador doméstico que se puede ir ampliando hasta obtener la potencia y prestación del *ML-G3.*

Este último puede dedicarse a realizar trabajos un poco más profesionales, aprovechando la capacidad y rapidez de la unidad de disco. En definitiva, se tratan de nuevas alternativas a la hora de comprar un ordenador.

El punto de discordia es el precio. El *ML-G1* saldrá al mercado por unas 79.910 ptas., y el *ML-G3* por unas 147.322 ptas., ambos sin IVA. Sin embargo, estos precios bajarán (según nos confirmó la casa) en un plazo bastante breve, por lo que la lucha en este sector del mercado va a ser muy importante.

Mitsubishi ha realizado un notable esfuerzo al desarrollarlos, y esto se nota en cuanto estamos frente a cualquiera de los dos ordenadores. Sus rivales más inmediatos, *VG-8235* y *HB-500P* son de temer hasta cierto punto, puesto que ambos ordenadores de Mitsubishi pueden castigar severamente a cualquiera de ellos y más cuando se trata del *ML-G3* del que sólo podemos decir CHAPEAU!

Foto 7. En el lateral derecho se encuentran los conectores para joysticks, el interface para cassette y el reset.



Entre las muchas aplicaciones de los ordenadores, una de las más interesantes es el tratamiento estadístico de grandes masas de datos. La estadística es una de las ramas más útiles de la matemática desde un punto de vista eminentemente práctico. Un tratamiento adecuado de la información puede permitirnos saber, por ejemplo, cuál será el candidato que tendrá más posibilidades de ganar las próximas elecciones, o cuántos días lloverá durante el próximo mes de diciembre.

La naturaleza no nos permite saber con absoluta certeza cuándo ocurrirá un determinado suceso o fenómeno, como por ejemplo la aparición de una nueva estrella, o conocer exactamente el lugar que ocupará un electrón determinado en un instante dado. Sin embargo, gracias a la estadística, podemos conocer el nacimiento de esa estrella o la posición del electrón con cierta precisión o, como se expresaría en física, con cierta incertidumbre. Así, podemos decir que la estrella aparecerá en el instante $T \pm \Delta T$ (es decir dentro de un intervalo de tiempo centrado en T y cuya anchura es $2 \Delta T$), o que el electrón estará en la posición p en el instante considerado.

Durante los próximos meses vamos a hablar de los fundamentos de la estadística, y daremos programas para el tratamiento estadístico de datos con ordendaores MSX.

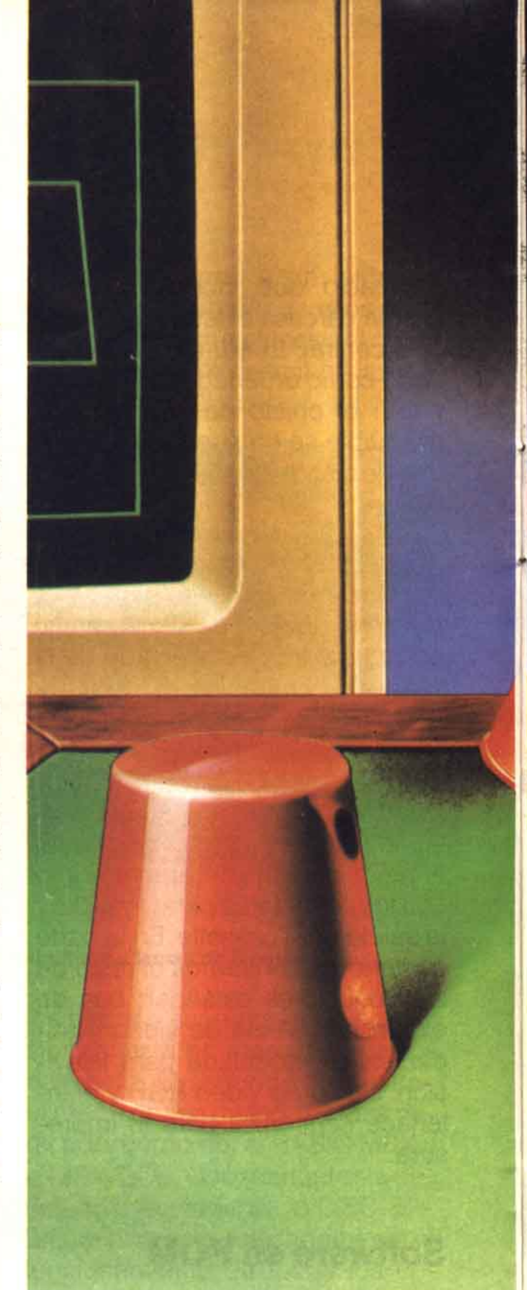
Comenzaremos definiendo los **parámetros estadísticos que definen una muestra de una variable.** Llamaremos **SUCESO** a un hecho concreto que ocurre al realizar un experimento. Por ejemplo, en el experimento «lanzar una moneda al aire y ver si sale cara o cruz» pueden ocurrir los dos sucesos: a) sale cara, y b) sale cruz.

POBLACION es el conjunto de elementos que cumplen una cierta propiedad o que tienen una característica común; y **MUESTRA** es una parte de dicho conjunto. Por ejemplo, podemos considerar la población: «todos los jóvenes de 15 años que hay en un colegio. Con esta muestra podemos considerar el suceso «ser rubio o rubia», y podremos realizar, con ella un análisis estadístico sobre el número de jóvenes de 15 años rubios o rubias que hay en la citada población. Así, estudiando un número reducido de elementos (muestra), podemos extraer conclusiones para todo el colectivo formado por elementos con propiedades análogas a las de los elementos de la muestra.

Para conocer los **parámetros estadísticos** que definen una muestra de una variable, supongamos el siguiente ejemplo: Vamos a estudiar el número de viajeros que utilizan una línea determinada de autobús a una hora fija del día. Para ello tomaremos una muestra que consistirá en contar, en 100 viajes que tengan lugar a esa hora, el número de viajeros. Así tendremos una muestra de tamaño 100. Y consideremos que los resultados de nuestras mediciones, ordenados de menor a mayor son los indicados en la tabla de la figura 1.

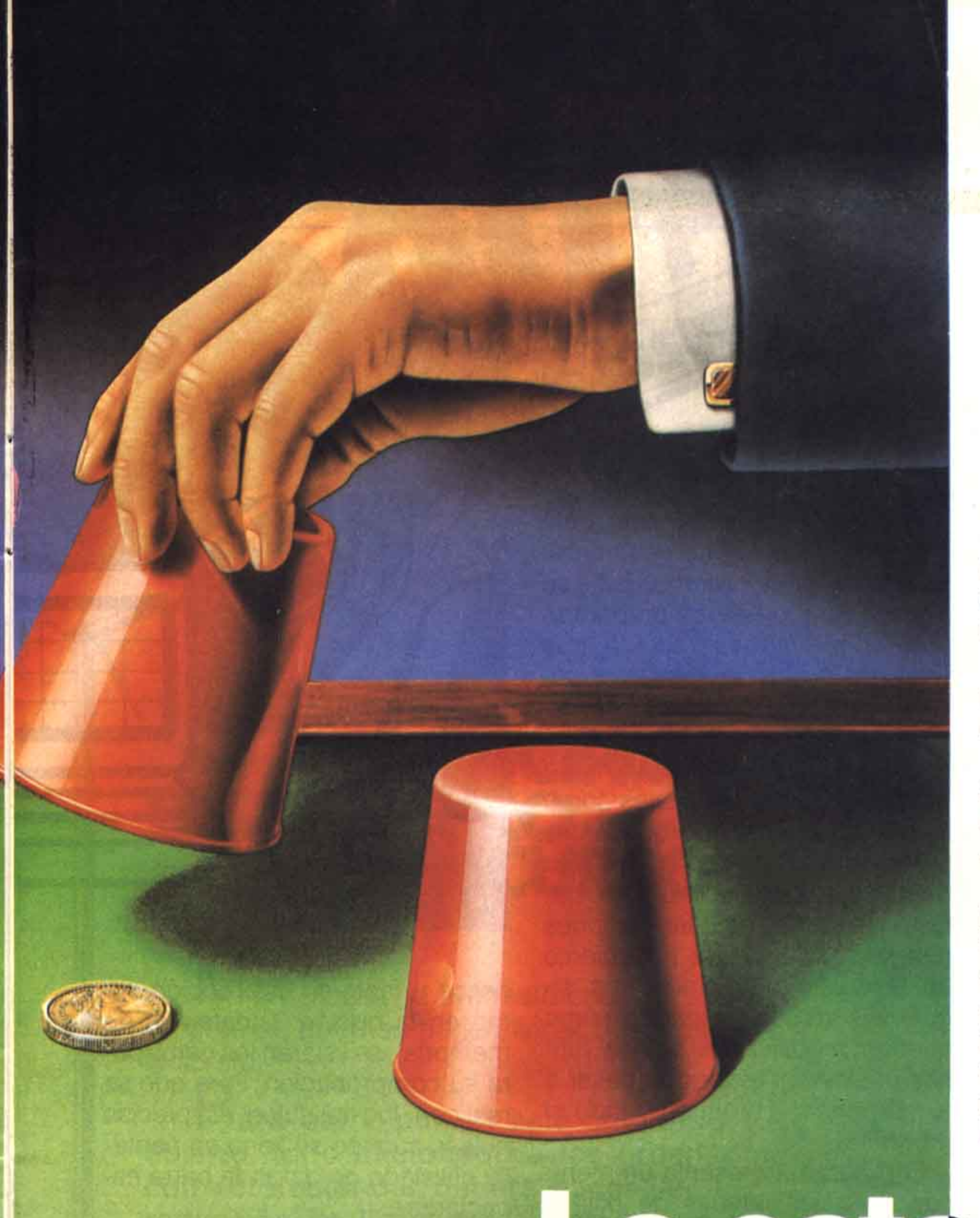
50	50	50	50	50	51	51	52	52	53
54	54	54	54	54	55	56	57	57	57
58	58	58	58	58	58	59	59	59	59
60	61	61	61	62	62	62	63	63	63
63	64	64	65	65	65	66	66	66	66
66	67	67	68	68	69	69	69	70	70
70	71	72	72	73	73	73	73	73	74
74	74	74	74	75	75	75	75	75	75
76	76	77	77	77	77	78	78	78	79
79	79	79	80	80	80	80	80	80	80

Fig. 1: Número de viajeros en un autobús. Tamaño de la muestra = 100.



De la simple observación de la tabla citada poco se puede decir. Unicamente que el número de viajeros que utilizan el autobús a esa hora oscila entre 50 y 80, y eso porque la tabla está ordenada. Vemos, por tanto, que el hecho de tener una serie de valores para la variable considerada, poco o nada nos dice sobre el comportamiento de la población.

Con la ayuda de la estadística podemos obtener una serie de **PARAMETROS** que nos permitirán conocer, con cierta precisión,



La estadística y el ordenador

los resultados de nuestra observación y extraer conclusiones respecto a la población. De estos parámetros, los de mayor interés son los siguientes:

El **RECORRIDO (R)** es la diferencia entre el valor más alto y el más bajo que toma la variable en estudio, en la muestra. Si denomiamos a la variable por x, el recorrido es:

$$R = X_{max} - X_{min}$$

Podemos tener así una idea de la extendión del intervalo en que se encuentra confinada la variable estudiada en la muestra.

La **MEDIA ($\bar{x}$)** es, quizá, el parámetro de mayor interés, y podemos definirla como el cociente entre la suma de los valores que toma la variable en la muestra y el tamaño es ésta:

$$\bar{X} = \frac{X_1 + X_2 + X_3 + \cdots + X_N}{N} = \frac{\sum_{i=1}^{N} X_i}{N}$$

Esta definición de media se corresponde bastante fielmente con la idea intuitiva de valor medio de un conjunto de datos.

La **MODA (m)** es el valor más frecuente de la variable, es decir, el que más se repite.

La **MEDIANA (M)** es el valor de x tal que el 50 por ciento de los casos sean inferiores a él y el 50% superiores.

La **DESVIACION TIPICA (σ)** es, junto con la media, uno de los estadígrados de mayor interés. Se define como la raíz cuadrada de la suma de los cuadrados de las diferencias entre cada uno de los valores que toma la variable y la media, dividida por el tamaño de la muestra; es decir:

$$\sigma = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \cdots + (x_N - \bar{x})^2}{N}} = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N}}$$

La desviación típica da una idea de la dispersión de la muestra. Supongamos, para entenderlo mejor, que estudiamos la edad de los hijos de una familia cualquiera. Tomemos dos muestras: En la primera hay 2 hijos de 6 y 8 años respectivamente. La media es de 7 años, y la desviación típica es 1. En la segunda muestra también hay 2 hijos, pero de 12 y 2 años. La media también es 7, pero la desviación típica es ahora 5, lo que refleja lo alejados que están los valores de la muestra de la media.

La desviación típica nos permitirá, más adelante, determinar la precisión con que podemos predecir un hecho concreto.

La **VARIANZA (σ^2)** es un estadígrafo muy importante utilizado, principalmente, en muestreo. Se define como el cuadrado de la desviación típica:

$$\sigma = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N}$$

Para el ejemplo que hemos considerado al principio, tenemos, por tanto, los siguientes valores de los parámetros definidos:

POBLACION: Número de viajeros que utilizan una línea de autobús a una hora determinada.

Tamaño de la muestra	*100*
Recorrido	*30*
Media	*66.36*
Moda	*80*
Mediana	*66*
Desviación típica	*9.298*
Varianza	*86.4504*

Notas al programa

El programa que listamos junto con este artículo tiene una numeración un tanto extraña. Esto se debe a que lo hemos preparado para modificarlo a medida que vayamos ampliando nuestros conocimientos de estadística.

En este programa hay una serie de rutinas que serán comunes a programas sucesivos, tales como la de lectura de datos procedentes del teclado o de cinta, la grabación de datos en cinta, el listado de datos o las rutinas de interrupciones (On key gosub) y de tratamiento de errores. En los números siguientes iremos indicando cómo quedan modificadas dichas rutinas para evitar tener que pasar a la memoria del ordenador todo el programa.

El programa presenta un menú con las posibilidades de utilizar datos procedentes del teclado, de cinta, o de reutilizar los datos que ya estén en la memoria del ordenador. Tras introducir los datos mediante el teclado, éstos se listarán para su comprobación, pudiendo ser modificados pulsando la tecla de función (F3). A continuación se presenta la posibilidad de grabar los datos en cinta, y acto seguido se calculan los parámetros que ya hemos visto. Si se opta por la opción 2 (datos de cinta), se leerán los datos de la cinta magnética y, tanto en esta opción como en la opción 3 (datos ya en memoria), se listarán los datos para su comprobación. Para que se realicen los cálculos es preciso pulsar, cuando se tiene en pantalla el listado de datos, la barra espaciadora.



En cualquier momento del programa es posible accionar el motor de la grabadora pulsando (F1) o regresar al menú pulsando (F2). En la línea 3200 se define una rutina en código máquina. Se trata de una rutina del sistema que nos permitirá (llamándola mediante *U=USR(0))* reestablecer la teclas de función una vez el programa se ha interrumpido al pulsar las teclas *CTRL+STOP*.

J. Antonio Feberero

```
100 ' PARAMETROS ESTADISTICOS
150 ' ======================
200 '
250 ' Juan Antonio Feberero Castejón
300 ' Versión 04.090686 - 4098 Bytes
350 '
1000 '
1050 'Inicialización
1100 '==============
1150 '
1200 SCREEN 0,,0
1250 WIDTH 39
1300 DEFINT I-K,N
1350 FOR I=4 TO 10
1400 KEY I,""
1450 NEXT I
1500 KEY 1,"motor"
1550 KEY 2,"menú"
```


GANE
7.000 PTAS.
todos los meses
PARTICIPANDO
EN NUESTRO
CONCURSO
MSX Magazine premiará cada mes los programas que nos hagan llegar nuestros lectores.
Para participar en este concurso abierto, todo aficionado a los ordenadores con este estándar deberá hacer llegar a la redacción de la revista el listado, un cassette y un texto explicativo.
Entre todos los programas que recibamos cada mes, serán seleccionados para su publicación aquellos que reúnan los siguientes criterios:
— Originalidad de la aplicación.
— Simplicidad del método de programación.
La única condición para participar en el concurso será que los programas no hayan sido publicados previamente en ninguna revista.
Enviar vuestros programas a: MSX Magazine
C/Bravo Murillo, 377 - 5.° A 28020 MADRID
BANCO DE ESPAÑA
MIL pesetas
CINCO MIL pesetas

```
1600 KEY 3,"modific"
1650 ON ERROR GOTO 55200
1700 ON KEY GOSUB 50200,50300,50400
1750 KEY(1) ON
1800 KEY(2) ON
1850 ON STOP GOSUB 60000
1900 STOP ON
3000 '
3050 'Rutina C.M. Rest. teclas función
3100 '--------------------------------
3150 '
3200 DEFUSR=&H3E
4000 '
4050 'Menú
4100 '====
4150 '
4200 LOCATE 8,5
4250 PRINT "PARAMETROS ESTADISTICOS"
4300 PRINT TAB(11);"PARA UNA VARIABLE"
4350 PRINT TAB(8);STRING$(23,195)
4400 LOCATE 8,10
4450 PRINT "1. Datos de teclado."
4500 PRINT
4550 PRINT TAB(8);"2. Datos de cinta."
4600 PRINT
4650 PRINT TAB(8);"3. Datos ya en memori
     a."
4700 WD$=INKEY$
4750 IF WD$="" THEN 4700
4800 IF INSTR("123",WD$)=0 THEN 4700
4850 CLS
4900 ON VAL(WD$) GOTO 6150,9080,10080
6000 '
6050 'Datos de teclado
6100 '================
6150 '
6200 INPUT "NUMERO DE DATOS";N
6250 PRINT
6300 ERASE X
6350 DIM X(N)
6400 FOR I=1 TO N
6450 PRINT USING"X(####):";I;
6500 INPUT X(I)
6550 NEXT I
6600 GOSUB 10080
6650 CLS
7000 '
7020 'Grabación de datos
7040 '------------------
7060 '
7080 PRINT "¿Deseas grabas los datos en
     cinta? S/N"
7100 W1$=INKEY$
7120 IF W1$="" THEN 7100
7140 IF INSTR("SNsn",W1$)=0 THEN 7100
7160 IF INSTR("Ss",W1$)=0 THEN 20200
7180 PRINT
7200 PRINT "Prepara la grabadora... [RET
     URN]."
7220 IF INKEY$<>CHR$(13) THEN 7220
7240 A$=""
7260 OPEN "DATOS" FOR OUTPUT AS#1
7280 PRINT "¿Deseas introducir comentari
     o? S/N"
7300 W1$=INKEY$
7320 IF W1$="" THEN 7300
7340 IF INSTR("SNsn",W1$)=0 THEN 7300
7360 IF INSTR("Ss",W1$)=0 THEN 7420
7380 PRINT "Introduce comentario (máx 25
     5 caracte- res...";
7400 LINE INPUT A$
7420 PRINT#1, A$
7440 PRINT#1,STR$(N)
7460 FOR I=1 TO N
7480 PRINT#1,STR$(X(I))
7500 NEXT I
7520 CLOSE #1
7540 CLS
7560 PRINT "Rebobina y pulsa [RETURN].",
     "Voy a comprobar la grabación."
7580 IF INKEY$<>CHR$(13) THEN 7580
7600 OPEN "DATOS" FOR INPUT AS#1
7620 PRINT
7640 PRINT "Comprobando..."
7660 LINE INPUT#1,AC$
7680 IF AC$<>A$ THEN 7860
7700 LINE INPUT#1,AC$
7720 IF AC$<>STR$(N) THEN 7860
7740 FOR I=1 TO N
7760 LINE INPUT#1,AC$
7780 IF AC$<>STR$(X(I)) THEN 7860
7800 NEXT I
7820 CLOSE #1
7840 GOTO 20200
7860 PRINT
7880 PRINT "¡¡¡ARCHIVO MAL GRABADO¡¡¡","
     INTENTALO DE NUEVO"
7900 GOTO 7180
9000 '
9020 'Datos de cinta
9040 '==============
9060 '
9080 ERASE X
9100 PRINT "Prepara la grabadora... [RET
     URN]."
9120 IF INKEY$<>CHR$(13) THEN 9120
9140 OPEN "DATOS" FOR INPUT AS#1
9160 PRINT "¿Deseas leer comentario? S/N
     "
9180 W1$=INKEY$
9200 IF W1$="" THEN 9180
9220 IF INSTR("SNsn",W1$)=0 THEN 9180
9240 LINE INPUT#1,A$
9260 IF INSTR("Ss",W1$)=0 THEN 9300
9280 PRINT A$
9300 LINE INPUT#1,A$
9320 N=VAL(A$)
9340 DIM X(N)
9360 FOR I=1 TO N
9380 LINE INPUT#1,A$
9400 X(I)=VAL(A$)
9420 NEXT I
9440 CLOSE #1
10000 '
10020 'Listado de datos
10040 '----------------
10060 '
10080 CLS
10100 FOR I=1 TO N
10120 KEY(3) STOP
10140 PRINT USING "X(####)=&";I;STR$(X(I
      ))
10160 IF I=N THEN 10200
10180 IF CSRLIN<20 THEN 10400
10200 KEY(3) ON
10220 W$=INKEY$
10240 IF W$="" THEN 10220
10260 IF W$=CHR$(30) THEN I=I-CSRLIN-20:
      GOTO 10340
10280 IF W$=CHR$(31) THEN 10360
10300 IF W$=CHR$(32) THEN IFWD$="1" THEN
      RETURN ELSE 20200
10320 GOTO 10220
10340 IF I<0 THEN I=0
10360 IF I=N THEN 10220
10380 CLS
10400 NEXT I
20000 '
20050 'Cálculos
20100 '========
20150 '
20200 CLS
20250 XI=X(1):XS=X(1)
20300 FOR I=2 TO N
20350 IF X(I)<XI THEN XI=X(I)
20400 IF X(I)>XS THEN XS=X(I)
20450 NEXT I
20500 CLS
20550 PRINT "X INFERIOR..:";XI
20600 PRINT
20650 PRINT "X SUPERIOR..:";XS
20700 PRINT
20750 PRINT "RECORRIDO...:";XS-XI
20800 XM=0:DT=0:MD=0
20850 FOR I=1 TO N
```

```
20900 XM=XM+X(I)
20950 NEXT I
21000 XM=XM/N
21050 PRINT
21100 PRINT "MEDIA.......:";XM
21150 FOR I=1 TO N
21200 DT=DT+(X(I)-XM)^2
21250 NEXT I
21300 DT=DT/N
21350 PRINT
21400 PRINT "DESV. TIPICA:";SQR(DT)
21450 PRINT
21500 PRINT "VARIANZA....:";DT
21550 FOR I=1 TO N
21600 MA=0
21650 FOR J=1 TO N
21700 IF X(J)=X(I) THEN MA=MA+1
21750 NEXT J
21800 IF MA>MO THEN MO=MA:IM=I
21850 NEXT I
21900 PRINT
21950 PRINT "MODA........:";X(IM)
22000 PRINT TAB(14);"FRECUENCIA..:";MO
22050 ME=N/2
22100 IF INT(ME)-ME<0 THEN ME=X(ME+1):GO
      TO 22200
22150 ME=(X(ME)+X(ME+1))/2
22200 PRINT
22250 PRINT "MEDIANA.....:";ME
22300 LOCATE ,23
22350 PRINT "Fin de cálculos - [F2] para
       menú.";
22400 GOTO 22400
50000 '
50050 'Rutinas On key gosub
50100 '--------------------
50150 '
50200 MOTOR        'Key 1:Motor
50250 RETURN       '-----------
50300 CLS          'Key 2:Menú
50350 RETURN 4200  '----------
50400 CS=CSRLIN    'Key 4:Modif. datos
50450 LOCATE 0,21 '------------------
50500 INPUT "Indice del dato a modificar
       ";IND
50550 PRINT USING "X(####)";IND;
50600 INPUT X(IND)
50650 LOCATE ,CS
50700 I=I+20
50750 W$=CHR$(30)
50800 RETURN 10260
55000 '
55050 'Errores
55100 '-------
55150 '
55200 IF ERR=5 AND ERL=6300 OR ERL=9080
       THEN RESUME NEXT
55250 IF ERR=6 THEN RESUME NEXT
55300 PRINT "Error ";ERR;" en linea ";ER
      L
60000 U=USR(0) 'On stop...Rest teclas
60050 KEY ON   '--------------------
60100 STOP
60150 GOTO 1350
```

Las instruccio
ocultas del Z8

El *chip Z80* de *ZILOG* trabaja con cerca de 700 instrucciones, usadas por los programadores de código máquina, de las que existe una amplia documentación. Hay también, unas 98 instrucciones ocultas que no son mencionadas por *ZILOG* y con las que no debes estar familiarizado, pero que parecen funcionar en cualquier procesador *Z80.*

Estas instrucciones extra son conocidas desde hace algunos años y no tiene ningún mérito el descubrirlas. Algunos escritores de programas de juegos las están usando, bien por conveniencia, o en sistemas de protección para confundir al «*hacker*» incauto. Debido a que la mayoría de los paquetes de ENSAMBLADORES y DESEMSAMBLADORES no las reconocen. Puede ser muy difícil encontrar la forma en que otros programadores las utilizan, o usarlas en vuestros propios programas.

¿Qué es lo que hacen y de donde han venido estos comandos extra? La mayoría conciernen a los registros índices IX e IY y hay uno de desplazamiento que parecía haber desaparecido. Si echamos un vistazo a la lista de comandos de rotación y desplazamientos excepto uno de ellos tiene versión para izquierda y derecha:

RCC	RRC
RL	RR
SLA	SRA
RLD	RRD
	SRL

La última instrucción *SRL* (desplazamiento lógico derecha) no tiene compañera, la cual debería ser *SLL* (desplazamiento lógico izquierda), y hay un espacio en la secuencia lógica de los códigos hexadecimales precisamente donde estas instrucciones deberían estar, entre *CB30* y *CB37.*

Experimentando con estos códigos se aprecia que las instrucciones son reconocidas y que un desplazamiento lógico a la izquierda tiene lugar. Pero no es exactamente como esperábamos. Mirando la figura se aprecia como, usando *SRL,* el registro es desplazado a la derecha un *bit;* el *bit* 0 pasa al banderín de acarreo y se coloca un 0 en el *bit* 7.

SLL debería hacer lo contrario, y lo hace, casi. Hay un desplazamiento de un *bit* a la izquierda en el registro, y el *bit* 7 pasa al banderín de acarreo. Pero en vez de poner 0 en el *bit* 0, pone un 1 (Figura 1).

Todos los banderines son colocados/inicializados como si hubiéramos ejecutado *SRL.* Así, una vez que conocemos como trabaja *SLL,* aunque no sea exactamente como esperamos, ¿por qué no utilizarlo?

No es un super-comando pero pudiera ser útil. La tabla 1 muestra todas las formas de *SLL* y sus códigos hexadecimales.

Veamos ahora el otro grupo de estas instrucciones ocultas, que se refieren a los registros *IX* e *IY.* Si estudiamos los códigos de los comandos que se refieren a uno de los registros índice y los comparamos con las instrucciones equivalentes del *HL,* veremos que los códigos del registro índice son los códigos de los comandos del *HL* precedidos por *DD* (si es *IX*) o *FD* (si es *IY*), con la adición del *byte* que indica el desplazamiento requerido (Figura 2).

Además, encontramos algunos huecos en la secuencia numérica de los códigos de las instrucciones de *IX* e *IY.* Por ejemplo, *LD A,H* (código 7C), no hay ningún equivalente para *IX* o *IY* publicado, sus códigos serían *DD 7C* o *FD 7C.*

Experimentando con esos códigos, encontraremos casi seguramente que el octeto superior de *IX* o *IY* es cargado en el acumulador. Similarmente, *DEC H* (código 2B), si es precedido por *DD,* decrementará la parte superior del re-

gistro *IX*, y precedido por *FD* decrementará la de *IY*. De repente tenemos otros cuatro registros de 8 *bits* disponibles, los octetos superior e inferior de *IX* e *IY*.

Todas estas nuevas instrucciones manejan los octetos individuales de los registros índices, así

ones
80

sus códigos de instrucción se relacionan con los de aquéllas que operen con los registros H y L. Nos referimos a estos octetos como *XH* (high x = octeto alto de *IX*) y *XL* (low x = octeto bajo de *IX*) para las de *IX* y como *YH* e *YL* para las del registro *IY*.

La lista completa de instrucciones extra y sus códigos se encuentra en la tabla 2. Cualquier cambio en los banderines resultante de la ejecución de los nuevos comandos será idéntico que el ocurrido para las instrucciones equivalentes de *H*, *L* o *HL*.

Aunque estas instrucciones extra no están reconocidas por *ZILOG* en su documentación del *Z80* no hay ninguna noticia sobre algún *Z80* en el que no hayan funcionado. Pero quizá pienses de forma diferente. Existe la posibilidad de que *ZILOG* pudiera fabricar una versión del *Z80* que no las acepte, pero como las modificaciones de un procesador tan popular serían muy costosas, es bastante improbable que esto suceda.

SLL (HL)	CB 36
SLL (IX+nn)	DD CB nn 36
SLL (IY+nn)	FD CB nn 36
SLL A	CB 37
SLL B	CB 30
SLL C	CB 31
SLL D	CB 32
SLL E	CB 33
SLL H	CB 34
SLL L	CB 35

TABLA 1. Formas del SLL y su códigos.

ADC A,XH	DD 8C	LD XH,A	DD 67
ADC A,XL	DD 8D	LD XH,B	DD 60
ADD A,XH	DD 84	LD XH,C	DD 61
ADD A,XL	DD 85	LD XH,D	DD 62
AND XH	DD A4	LD XH,E	DD 63
AND XL	DD A5	LD XL,A	DD 6F
CP XH	DD BC	LD XL,B	DD 68
CP XL	DD BD	LD XL,C	DD 69
DEC XH	DD 25	LD XL,D	DD 6A
DEC XL	DD 2D	LD XL,E	DD 6B
INC XH	DD 24	LD XL,XH	DD 6C
INC XL	DD 2C	LD XL,nn	DD 2E nn
LD A,XH	DD 7C	LD XH,XL	DD 65
LD A,XL	DD 7D	LD XH,nn	DD 26 nn
LD B,XH	DD 44	OR XH	DD B4
LD B,XL	DD 45	OR XL	DD B5
LD C,XH	DD 4C	SBC A,XH	DD 9C
LD C,XL	DD 4D	SBC A,XL	DD 9D
LD D,XH	DD 54	SUB XH	DD 94
LD D,XL	DD 55	SUB XL	DD 95
LD E,XH	DD 5C	XOR XH	DD AC
LD E,XL	DD 5D	XOR XL	DD AD

TABLA 2. Las instrucciones ocultas de los registros índices. XH y XL son los octetos alto y bajo de IX respectivamente. Para los comandos del registro IY sustituir DD por FD.

Este programa permite listar en mnemónicos cualquier programa en código máquina *X80* y hacer volcados de memoria en hexadecimal, *ASCII* o ambos a la vez. Es compatible con disco e impresora; la salida puede obtenerse por pantalla o impresora de 80 columnas. Cambiando dos líneas puede usarse en los ordenadores *SVI-318/328;* esto es muy interesante para los usuarios de estas máquinas ya que, de momento, no hay ningún programa comercial de este tipo para las mismas.

Cómo introducir el programa

Es conveniente redefinir algunas teclas de función para facilitar el trabajo. Teclear lo que sigue en modo directo.

```
KEY1,"MN$="CHR$(&H22)
KEY2,"TEHN"
KEY4,"GOSUB"
KEY5,"RETURN"+CHR$(&H0D)
```

Es tambie aprovechable la F3 *("GOTO").* Para devolver a las teclas sus funciones normales, introducir *DEFUSR*= 62:*?USR*(0):*CLS.* Esto es una llamada a la rutina *ROM* que asigna las funciones iniciales. La numeración de líneas comienza en 10 y aumenta en saltos de 10. Por tanto, para obtener cómodamente los números de línea teclear *AUTO* y *RETURN;* a partir de aquí sólo hay que introducir los textos de las líneas.

Si se quiere quitar los *REM* basta con pulsar *RETURN* al aparecer el correspondiente número de línea. Haciendo esto se tendrán unos 700 *bytes* más disponibles.

Características

Por estar escrito en *BASIC,* el programa se instala en la parte baja de la *RAM* del usuario (Comienzo en 32768 (&H8000) para 32 ó más *Kbytes* de *RAM* y 49152 (&HC000) para 16 *Kbytes* de *RAM*).

Al trabajar no habrá problema si la parte que interesa estudiar está en *ROM* (direcciones / - 32767). Si el código máquina debe estar en *RAM* hay que cargarlo en alguna zona donde el *BASIC* no pueda alterarlo accidentalmente. La variable del sistema *MEMSIZ* (*MEMory SIZe* = tamaño de memoria) da una dirección que divide la *RAM* en dos partes. La parte más baja es la llamada «área del usuario»; es administrada por el intérprete de *BASIC,* que la utiliza según sus necesidades. La parte alta contiene la zona de variables del sistema, que tiene una longitud fija de 2586 *bytes,* a los que hay que añadir 4895 más si se está usando el *BASIC* de disco; el espacio existente entre *MEMSIZ* y la primera dirección de la zona del sistema queda libre de acceso por parte del *BASIC* y es un lugar seguro para guardar el código máquina.

Por medio de la instrucción *CLEAR* a,b puede variarse la dirección a la que apunta *MEMSIZ;* «a» es el espacio asignado para cadenas de caracteres y «b» es el nuevo *"MEMSIZ".* A continuación se dan los valores mínimos que pueden emplearse con *CLEAR.* Valores más pequeños pueden dar lugar a un error *"OUT OF STRING SPACE"* para «a» y *"OUT OF MEMORY"* para «b».

32 ó más *Kbytes* de *RAM*

CLEAR 100,41854
(desensamblador sin *REMs*)

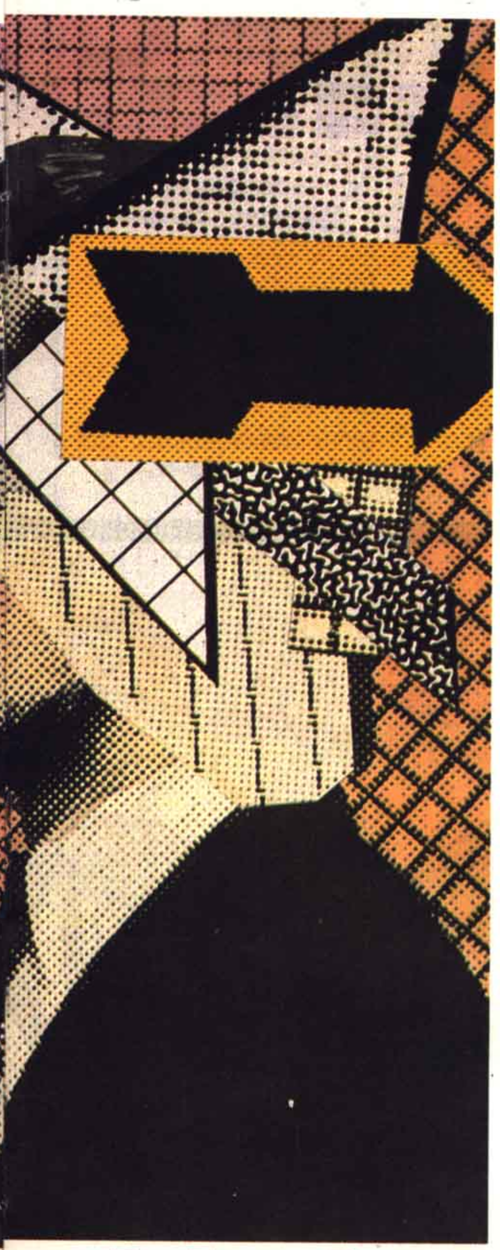


Desensa

CLEAR 100,42604
(desensamblador con *REMs*)

16 *Kbytes* de *RAM*

CLEAR 100,58238
(desensamblador sin *REMs*)



mblador

CLEAR 100,58988
(desensamblador con *REMs*)

Teniendo en cuenta que el comienzo de la zona de trabajo está en 62336 con cassette y 57441 con diskette tenemos la siguiente tabla que nos da las posibles longitudes de código al desensamblar:

32 K RAM

con *REMs*	
cassette	diskette
19732	14837
sin *REMs*	
cassette	diskette
20482	15587

16 K RAM

con *REMs*	
cassette	diskette
3348	n/a
sin *REMs*	
cassette	diskette
4098	n/a

No se deben sobrepasar estos límites, ya que se invadiría la zona de variables del sistema con lo que, muy probablemente, quedaría bloqueado el ordenador y habría que efectuar un *RESET* para recuperar el control.

Relocalización de programas

A pesar de los límites expuestos, es posible desensamblar cualquier programa en código máquina, no importa cual sea su longitud o dirección de comienzo. Para ello usaremos las facilidades que nos da la instrucción *BLOAD* y una característica de este programa, la RELOCALIZACION o posibilidad de desensamblar un programa cargándolo en una dirección distinta de la de ejecución.

Para trabajar con un programa largo, que no quepa en la zona que hayamos dejado libre con *CLEAR*, daremos estos pasos:

1. Dividir el programa en bloques suficientemente cortos.

2. Salvar cada bloque con la instrucción
BSAVE"dispositivo:nombre", inicio,fin

3. Para desensamblar, cargar cada bloque separadamente.

Al cargar un programa, o un bloque separado por el sistema anterior, es posible que todo o parte de él quede fuera de zona de trabajo del desensamblador. Para «relocalizarlo» se hace lo siguiente:

1.° Cargarlo mediante la instrucción *BLOAD*"dispositivo:nombre", desplazamiento. Esta cantidad es el número de *bytes* a sumar o restar de la dirección con que fue grabado el código, para obtener la dirección de carga. Si es negativo hay que sumarle 65536 para obtener un número positivo.

2.° Al ejecutar el programa desensamblador, introducir las direcciones de comienzo actual en memoria y de comienzo real.

Usando el programa

Los datos que pide el programa (direcciones) y los que proporciona el desensamblador (constantes y direcciones) se dan en valores decimales. Resulta fácil modificar el programa para trabajar sólo en hexadecimales. El programa se usa de esta forma:

1. Cargar el desensamblador.

2. Efectuar el *CLEAR* más conveniente.

3. Cargar el código máquina.

4. Teclear *RUN* y *ENTER*. Aparecen los siguientes mensajes:

"Dirección inicial?". Dirección REAL de comienzo del programa.

"Dirección final?". Dirección hasta la que se va a desensamblar.

"Coinciden las direcciones reales con las actuales en memoria?". Si se contesta «n» aparece el mensaje "Dirección inicial en MEMORIA?". Introducir la dirección a partir de donde se ha cargado el código.

"Salida por impresora? s/n". Pulsar a voluntad.

5. Elegir una de las siguientes opciones del menú:

1.ª Listado hexadecimal. El formato es en líneas de 0 bytes (24 = 3x8 en impresora) encabezadas por la dirección real del primer byte.

2.ª Listado *ASCII*. Como el anterior, pero cada *byte* está representado por un código *SCII*. Los valores mayores de 127 (caracteres especiales) y menores de 32 (códigos de control) se representan por puntos (.). Los ceros, por ser muy característicos se representan con un signo.

3.ª Listado combinado. Reune los dos anteriores. En cada línea se representan 8 *bytes* en pantalla y 16 en impresora.

4.ª Listado en mnemónicos. Está formado por:

```
56   38  ·JP 3182   C33C03c
a     b     c     d     e
```

a) Dirección del primer *byte* de la instrucción en decimal.
b) Dirección del primer *byte* de la instrucción de hexadecimal. Se presenta sólo en impresora.
c) Mnemónicos de la instrucción.
d) Códigos hexadecimales de la instrucción.
e) Códigos *ASCII* de la instrucción.

Todas las opciones terminan con la vuelta a modo comando (mensaje «OK» en pantalla).

A no ser que se tenga una idea clara de la estructura del programa a desensamblar no debe comenzarse con la opción 4. En código máquina no hay forma, inicialmente, de diferenciar datos de instrucciones; hace falta, por tanto, un estudio previo. La opción 3 (y también las 1 y 2) dan una mayor visión de conjunto y ayudan a descubrir los mensajes *ASCII* que pueda contener el programa. Probar a introducir como dirección de comienzo 15734; la opción 4 da una serie de mnemónicos válidos pero, si usamos la opción 3, vemos que lo que estamos desensamblando no son instrucciones y que lo que aparece son los mensajes de error del *BASIC*.

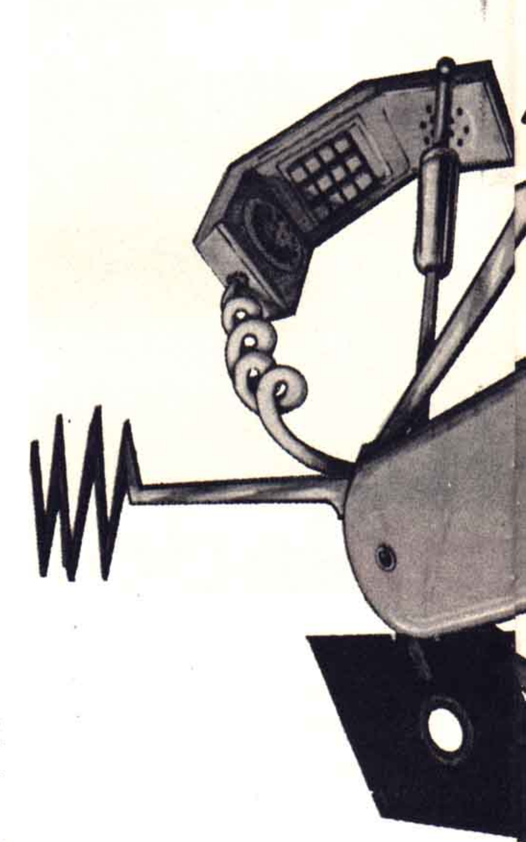
Adaptación a Spectravídeo 318/328

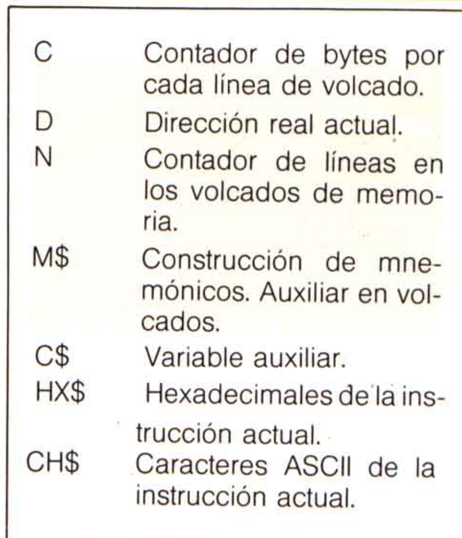
Este programa es perfectamente válido para dichos ordenadores. Deben efectuarse al teclearlo los cambios siguientes:

```
Línea 70 SCREEN,0:LOCATE,,0:
         WIDTH(40):CLS
Línea 380 LOCATE,,1:END
```

Variables del desensamblador

PRG	Dirección real de comienzo en código máquina.
FIN	Dirección final.
LNG	Longitud del código, en bytes.
DIR	Dirección de carga, para el desensamblado.
DF	Diferencia dirección real – dirección de desensamblado.
HX	Dirección actual del desensamblado.
CT	Contador.
RP	Bandera. 0=salida por pantalla. 1=impresora.
B	Valor hexadecimal del byte.





C	Contador de bytes por cada línea de volcado.
D	Dirección real actual.
N	Contador de líneas en los volcados de memoria.
M$	Construcción de mnemónicos. Auxiliar en volcados.
C$	Variable auxiliar.
HX$	Hexadecimales de la instrucción actual.
CH$	Caracteres ASCII de la instrucción actual.

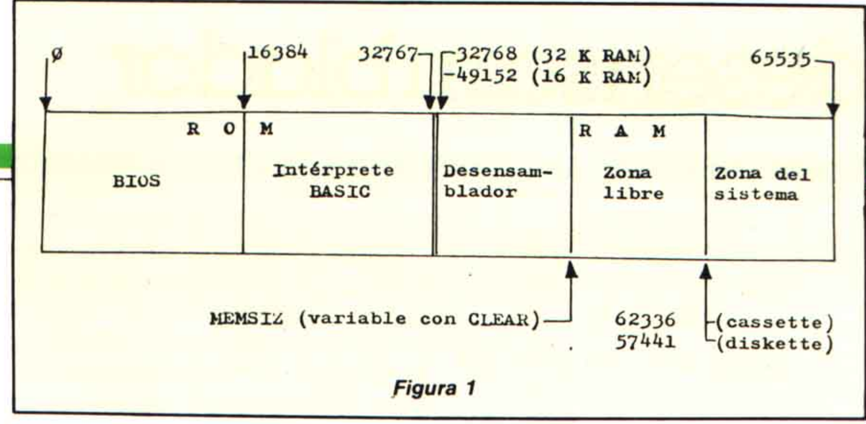


Figura 1

Estructura del programa

El desarrollo es muy sencillo. Tiene dos partes: una de inicialización, donde el usuario introduce los datos necesarios, y una de ejecución. Esta última está formada por siete secciones independientes; cada una tiene una sola salida y una sola entrada.

La parte del desensamblador, aunque extensa, es poco complicada. Cada instrucción es decodificada por una rutina propia, apareciendo todas en la misma secuencia que la de las tablas de instrucciones del *Z80*. Se ha preferido esta disposición a incluir los nombres en datas, ya que esto último hace necesario pasar los «datas» a variables, suponiendo esto una pérdida de tiempo al principio del programa y una menor claridad en el listado.

Para hacer la decodificación se aprovecha la gran racionalidad con que están distribuidas las instrucciones. Por ejemplo, las instrucciones con códigos de operación de un *byte* pueden agruparse en cuatro bloques perfectamente diferenciados. El segundo de éstos contiene las instrucciones de carga de registros, mientras que el tercero contiene las operaciones aritméticas y lógicas de un *byte*. Centrémonos en una instrucción cuyo código es 10001010 ó, lo que es igual, 138 (&H8A); ¿cómo aislarla de las restantes?

Separemos los 2 primeros bits; esta operación se llama «enmascarar» y se efectúa con un *AND*. En la máscara pondremos a «1» los bits que serán «transparentes» y a «0» a los que deban «tapar» al código en estudio; por tanto, la máscara será aquí 11000000.

desensamblador

```
        10001010   138
AND     11000000   192
        10000000   128
```

Dividiendo este resultado por 64 nos da 2. Numerando de 0 a 3 los grupos de que hablábamos anteriormente vemos que esta instrucción corresponde al tercer grupo (operaciones aritmético-lógicas). Aislemos ahora los siguientes 3 *bits;* la máscara será ahora 00111000 = 56 (&H38).




```
        10001010   138
AND     00111000    56
        00001000     8
```



Consultando una tabla de instrucciones del Z80 vemos que la combinación XX001000 corresponde a la instrucción ADC A,... Para saber cuál es el registro afectado, aislemos los 3 últimos bytes; la máscara será 00000111, o sea, 7.

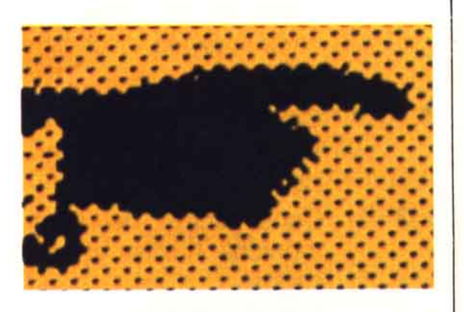
```
        10001010   138
AND     00000111     7
        00000010     2
```

La combinación XXXXX010 corresponde al registro D. Este proceso nos da por resultado.

138 = ADC A,D

Este tipo de operaciones es la base de todo el programa.

En cuanto a los volcados, cada



una de las 6 partes es totalmente independiente y puede separarse del programa principal, si se desea más espacio. Todas tienen la misma estructura: un bucle que divide el listado en líneas (variable N) y otro que recorre cada línea (variable C).

Lorenzo Hernández Talavera
Cádiz

```
10 '#################################
20 '#                               #
30 '#      desensamblador Z-80      #
40 '#   (c)86 Lorenzo H. Talavera   #
50 '#                               #
60 '#################################
70 KEYOFF:CLS:WIDTH(40)
80 STOPON:ONSTOPGOSUB380
90 INPUT"Direccion inicial";PRG
100 INPUT"Direccion final";FIN:LNG=FI
   N-PRG
110 PRINT:PRINT"Coinciden las direcci
   ones reales con lasactuales en m
   emoria?"
120 M$=INKEY$:IFM$=""THEN120ELSEIFM$=
   "N"ORM$="n"THENINPUT"Direccion i
   nicial en MEMORIA";DIRELSEDIR=PR
   G
130 PRINT:PRINT"Salida por impresora?
    s/n":PRINT:PRINT
140 M$=INKEY$:IFM$=""THEN140ELSEIFM$=
   "s"ORM$="S"THENRP=1ELSERP=0
150 PRINT"1.- Listado Hexadecimal":PR
   INT
160 PRINT"2.- Listado ASCII":PRINT
170 PRINT"3.- Listado combinado":PRIN
   T
180 PRINT"4.- Listado en mnemonicos":
   PRINT:PRINT"Elegir una opcion"
190 M$=INKEY$:IFM$<"1"ORM$>"4"THEN190
200 CLS:CT=DIR:DF=PRG-DIR:PRG=PRG-DF:
   FIN=FIN-DF
210 IFRP=1THENPRINT"SALIDA POR IMPRES
   ORA":ONVAL(M$)GOTO3040,3120,3200
220 ONVAL(M$)GOTO2730,2810,2900
230 '
240 '******  DESENSAMBLADOR  ******
250 '
260 B=PEEK(CT):HX=CT:D=CT+DF
270 ON((BAND192)/64)+1GOSUB400,840,88
   0,970
280 HX$="":FORN=HXTOCT
290 HX$=HX$+RIGHT$("0"+HEX$(PEEK(N)),
   2)
300 NEXTN
310 CH$="":FORN=HXTOCT:B=PEEK(N)
320 IFB=0THENC$="##"ELSEIFB>31ANDB<12
   8THENC$=CHR$(B)ELSEC$="."
330 CH$=CH$+C$
340 NEXTN
350 PRINTUSING"#####";D;:PRINT"    "+M
   $;:PRINTTAB(25)HX$;:PRINTTAB(35)
   CH$
360 IFRP=1THENLPRINTUSING"#####";D;:L
   PRINT"    "+RIGHT$("    "+HEX$(D),
   4)+"    "+M$;:LPRINTTAB(37)HX$;:L
   PRINTTAB(52)CH$
370 CT=CT+1:IFCT<=(DIR+LNG-1)THEN260
380 END
390 '     INSTRUCCIONES DE 1 BYTE
400 IF(BMOD8)=4THENM$="INC ":GOTO870
410 IF(BMOD8)=5THENM$="DEC ":GOTO870
420 IF(BMOD8)=6THENM$="LD ":GOSUB870:
```

```
M$=M$+",":GOTO2640
430 ONB+1GOTO440,450,460,470,,,,480,4
    90,500,510,520,,,,530,540,550,56
    0,570,,,,580,590,600,610,620,,,,
    630,640,650,660,670,,,,680,690,7
    00,710,720,,,,730,740,750,760,77
    0,,,,780,790,800,810,820,,,,830
440 M$=" ":RETURN
450 M$="LD BC,":GOTO2650
460 M$="LD BC,A":RETURN
470 M$="INC BC":RETURN
480 M$="RLCA":RETURN
490 M$="EX AF,AF'":RETURN
500 M$="ADD HL,BC":RETURN
510 M$="LD A,(BC)":RETURN
520 M$="DEC BC":RETURN
530 M$="RRCA":RETURN
540 M$="DJNZ":GOTO2660
550 M$="LD DE,":GOTO2650
560 M$="LD (DE),A":RETURN
570 M$="INC DE":RETURN
580 M$="RLA":RETURN
590 M$="JR ":GOTO2660
600 M$="ADD HL,DE":RETURN
610 M$="LD A,(DE)":RETURN
620 M$="DEC DE":RETURN
630 M$="RRA":RETURN
640 M$="JR NZ,":GOTO2660
650 M$="LD HL,":GOTO2650
660 M$="LD (":GOSUB2650:M$=M$+"),HL":
    RETURN
670 M$="INC HL":RETURN
680 M$="DAA":RETURN
690 M$="JR Z,":GOTO2660
700 M$="ADD HL,HL":RETURN
710 M$="LD HL,(":GOSUB2650:M$=M$+")":
    RETURN
720 M$="DEC HL":RETURN
730 M$="CPL":RETURN
740 M$="JR NC,":GOTO2660
750 M$="LD SP,":GOTO2650
760 M$="LD (":GOSUB2650:M$=M$+"),A":R
    ETURN
770 M$="INC SP":RETURN
780 M$="SCF":RETURN
790 M$="JR C,":GOTO2660
800 M$="ADD HL,SP":RETURN
810 M$="LD A,(":GOSUB2650:M$=M$+")":R
    ETURN
820 M$="DEC SP":RETURN
830 M$="CCF":RETURN
840 IFB=118THENM$="HALT":RETURN
850 IFB>127THEN880
860 M$="LD ":GOSUB870:M$=M$+",":GOTO1
    590
870 ON(BAND56)/8+1GOSUB1600,1610,1620
    ,1630,1640,1650,1660,1670:RETURN
880 ON(BAND56)/8+1GOTO890,900,910,920
    ,930,940,950,960
890 M$="ADD A,":GOTO1590
900 M$="ADC A,":GOTO1590
910 M$="SUB ":GOTO1590
920 M$="SBC A,":GOTO1590
930 M$="AND ":GOTO1590
940 M$="XOR ":GOTO1590
950 M$="OR ":GOTO1590
960 M$="CP ":GOTO1590
970 ON(BMOD8)+1GOTO980,1020,990,1020,
    1000,1020,1020,1010
980 M$="RET ":GOTO1030
990 M$="JP ":GOSUB1030:M$=M$+",":GOTO
    2650
1000 M$="CALL ":GOSUB1030:M$=M$+",":G
    OTO2650
1010 M$="RST"+STR$(BAND56):RETURN
1020 ONB-191GOTO,1120,,1130,,1140,115
    0,,,1160,,1170,,1180,1190,,,1200
    ,,1210,,1220,1230,,,1240,,1250,,
    1260,1270,,,1280,,1290,,1300,131
    0,,,1320,,1330,,1340,1350,,,1360
    ,,1370,,1380,1390,,,1400,,1410,,
    1420,1430
1030 ON(BAND56)/8+1GOTO1040,1050,1060
    ,1070,1080,1090,1100,1110
1040 M$=M$+"NZ":RETURN
1050 M$=M$+"Z":RETURN
1060 M$=M$+"NC":RETURN
1070 M$=M$+"C":RETURN
1080 M$=M$+"PO":RETURN
1090 M$=M$+"PE":RETURN
1100 M$=M$+"P":RETURN
1110 M$=M$+"M":RETURN
1120 M$="POP BC":RETURN
1130 M$="JP ":GOTO2650
1140 M$="PUSH BC":RETURN
1150 M$="ADD A,":GOTO2640
1160 M$="RET":RETURN
1170 CT=CT+1:GOTO1450
1180 M$="CALL ":GOTO2650
1190 M$="ADC A,":GOTO2640
1200 M$="POP DE":RETURN
1210 M$="OUT (":GOSUB2640:M$=M$+")":R
    ETURN
1220 M$="PUSH DE":RETURN
1230 M$="SUB ":GOTO2640
1240 M$="EXX":RETURN
1250 M$="IN A,":GOTO2640
1260 I$="IX":CT=CT+1:GOTO2190
1270 M$="SBC A,":GOTO2640
1280 M$="POP HL":RETURN
1290 M$="EX (SP),HL":RETURN
1300 M$="PUSH HL":RETURN
1310 M$="AND ":GOTO2640
1320 M$="JP (HL)":RETURN
1330 M$="EX DE,HL":RETURN
1340 CT=CT+1:GOTO1690
1350 M$="XOR ":GOTO2640
1360 M$="POP AF":RETURN
1370 M$="DI":RETURN
1380 M$="PUSH AF":RETURN
1390 M$="OR ":GOTO2640
1400 M$="LD SP,HL":RETURN
1410 M$="EI":RETURN
1420 I$="IY":CT=CT+1:GOTO2190
```

desensamblador

```
1430 M$="CP ":GOTO2640
1440 '       INSTRUCCIONES "CB"
1450 B=PEEK(CT)
1460 ON(BAND192)/64+1GOTO1470,1560,15
     70,1580
1470 ON(BAND56)/8+1GOTO1480,1490,1500
     ,1510,1520,1530,1540,1550
1480 M$="RLC ":GOTO1590
1490 M$="RRC ":GOTO1590
1500 M$="RL  ":GOTO1590
1510 M$="RR  ":GOTO1590
1520 M$="SLA ":GOTO1590
1530 M$="SRA ":GOTO1590
1540 M$="Error":RETURN
1550 M$="SRL ":GOTO1590
1560 M$="BIT"+STR$((BAND56)/8)+",":GO
     TO1590
1570 M$="RES"+STR$((BAND56)/8)+",":GO
     TO1590
1580 M$="SET"+STR$((BAND56)/8)+","
1590 ON(BAND7)+1GOTO1600,1610,1620,16
     30,1640,1650,1660,1670
1600 M$=M$+"B":RETURN
1610 M$=M$+"C":RETURN
1620 M$=M$+"D":RETURN
1630 M$=M$+"E":RETURN
1640 M$=M$+"H":RETURN
1650 M$=M$+"L":RETURN
1660 M$=M$+"(HL)":RETURN
1670 M$=M$+"A":RETURN
1680 '       INSTRUCCIONES "ED"
1690 B=PEEK(CT):M$="":IFB<64OR(B>123A
     NDB<160)ORB>187THEN2170
1700 IFB>123THEN2010
1710 IFBMOD8=0THENONB/8-7GOSUB1600,16
     10,1620,1630,1640,1650,1660,1670
     :M$="IN  "+M$+",(C)":RETURN
1720 IFBMOD8=1THENONB/8-7GOSUB1600,16
     10,1620,1630,1640,1650,1660,1670
     :M$="OUT (C),"+M$:RETURN
1730 IFB=66THENM$="SBC HL,BC":RETURN
1740 IFB=67THENM$="LD  (":GOSUB2650:M$
     =M$+"),BC":RETURN
1750 IFB=68THENM$="NEG":RETURN
1760 IFB=69THENM$="RETN":RETURN
1770 IFB=70THENM$="IM 0":RETURN
1780 IFB=71THENM$="LD I,A":RETURN
1790 IFB=74THENM$="ADC HL,BC":RETURN
1800 IFB=75THENM$="LD BC,(":GOSUB2650
     :M$=M$+")":RETURN
1810 IFB=77THENM$="RETI":RETURN
1820 IFB=79THENM$="LD R,A":RETURN
1830 IFB=82THENM$="SBC HL,DE":RETURN
1840 IFB=83THENM$="LD  (":GOSUB2650:M$
     =M$+"),DE":RETURN
1850 IFB=86THENM$="IM 1":RETURN
1860 IFB=87THENM$="LD A,I":RETURN
1870 IFB=90THENM$="ADC HL,DE":RETURN
1880 IFB=91THENM$="LD DE,(":GOSUB2650
     :M$=M$+")":RETURN
1890 IFB=94THENM$="IM 2":RETURN
1900 IFB=95THENM$="LD A,R":RETURN
1910 IFB=98THENM$="SBC HL,HL":RETURN
1920 IFB=99THENM$="LD  (":GOSUB2650:M$
     =M$+"),HL":RETURN
1930 IFB=103THENM$="RRD":RETURN
1940 IFB=106THENM$="ADC HL,HL":RETURN
1950 IFB=107THENM$="LD HL,(":GOSUB265
     0:M$=M$+")":RETURN
1960 IFB=111THENM$="RLD":RETURN
1970 IFB=114THENM$="SBC HL,SP":RETURN
1980 IFB=115THENM$="LD  (":GOSUB2650:M
     $=M$+"),SP":RETURN
1990 IFB=122THENM$="ADC HL,SP":RETURN
2000 IFB=123THENM$="LD SP,(":GOSUB265
     0:M$=M$+")":RETURN
2010 IFB=160THENM$="LDI":RETURN
2020 IFB=161THENM$="CPI":RETURN
2030 IFB=162THENM$="INI":RETURN
2040 IFB=163THENM$="OUTI":RETURN
2050 IFB=168THENM$="LDD":RETURN
2060 IFB=169THENM$="CPD":RETURN
2070 IFB=170THENM$="IND":RETURN
2080 IFB=171THENM$="OUTD":RETURN
2090 IFB=176THENM$="LDIR":RETURN
2100 IFB=177THENM$="CPIR":RETURN
2110 IFB=178THENM$="INIR":RETURN
2120 IFB=179THENM$="OTIR":RETURN
2130 IFB=184THENM$="LDDR":RETURN
2140 IFB=185THENM$="CPDR":RETURN
2150 IFB=186THENM$="INDR":RETURN
2160 IFB=187THENM$="OTDR":RETURN
2170 M$="Error":RETURN
2180 '   INSTRUCCIONES REGISTROS INDIC
     E
2190 B=PEEK(CT)
2200 IFB=9THENM$="ADD  "+I$+",BC":RETU
     RN
2210 IFB=25THENM$="ADD  "+I$+",DE":RET
     URN
2220 IFB=33THENM$="LD  "+I$+",":GOTO26
     50
2230 IFB=34THENM$="LD  (":GOSUB2650:M$
     =M$+"),"+I$:RETURN
2240 IFB=35THENM$="INC  "+I$:RETURN
2250 IFB=41THENM$="ADD  "+I$+",HL":RET
     URN
2260 IFB=42THENM$="LD  "+I$+",(":GOSUB
     2650:M$=M$+")":RETURN
2270 IFB=43THENM$="DEC  "+I$:RETURN
2280 IFB=52THENM$="INC  ":GOTO2610
2290 IFB=53THENM$="DEC  ":GOTO2610
2300 IFB=54THENM$="LD  ":GOSUB2610:CT=
     CT+1:M$=M$+","+STR$(PEEK(CT)):RE
     TURN
2310 IFB=57THENM$="ADD  "+I$+",SP":RET
     URN
2320 IFB>111ANDB<120THENM$="LD  ":GOSU
     B2610:M$=M$+",":ON(BAND7)+1GOTO1
     600,1610,1620,1630,1640,1650,247
     0,1670
2330 IFB>69ANDB<127ANDBMOD8=6THENM$="
     LD  ":ON(BAND56)/8+1GOSUB1600,161
     0,1620,1630,1640,1650,2470,1670:
     M$=M$+",":GOTO2610
```

```
2340 IFB=134THENM$="ADD ":GOTO2610
2350 IFB=142THENM$="ADC ":GOTO2610
2360 IFB=158THENM$="SBC ":GOTO2610
2370 IFB=166THENM$="AND ":GOTO2610
2380 IFB=174THENM$="XOR ":GOTO2610
2390 IFB=182THENM$="OR ":GOTO2610
2400 IFB=190THENM$="CP ":GOTO2610
2410 IFB=225THENM$="POP "+I$:RETURN
2420 IFB=227THENM$="EX (SP),"+I$:RETU
     RN
2430 IFB=229THENM$="PUSH "+I$:RETURN
2440 IFB=233THENM$="JP ("+I$+")":RETU
     RN
2450 IFB=249THENM$="LD SP,"+I$:RETURN
2460 IFB=203THENCT=CT+2:IF(PEEK(CT)MO
     D8)=6THENON(PEEK(CT)AND192)/64+1
     GOTO2480,2560,2570,2580
2470 M$="Error":RETURN
2480 ON(PEEK(CT)AND56)/8+1GOTO2490,25
     00,2510,2520,2530,2540,2470,2550
2490 M$="RLC ":GOTO2600
2500 M$="RRC ":GOTO2600
2510 M$="RL ":GOTO2600
2520 M$="RR ":GOTO2600
2530 M$="SLA ":GOTO2600
2540 M$="SRA ":GOTO2600
2550 M$="SRL ":GOTO2600
2560 M$="BIT ":GOTO2590
2570 M$="RES ":GOTO2590
2580 M$="SET "
2590 M$=M$=M$+STR$((PEEK(CT)AND56)/8)
     +","
2600 M$=M$+"("+I$+"+":B=PEEK(CT-1):GO
     SUB2620:M$=M$+")":RETURN
2610 CT=CT+1:M$=M$+"("+I$+"+":B=PEEK(
     CT):GOSUB2620:M$=M$+")":RETURN
2620 B=((B>127)*(256-B)-(B<128)*B):IF
     B<0THENM$=M$+STR$(B)ELSEM$=M$+"+
     "+RIGHT$(STR$(B),LEN(STR$(B))-1)
     :RETURN
2630 '   SUBRUTINAS DE CONVERSION
2640 CT=CT+1:M$=M$+STR$(PEEK(CT)):RET
     URN
2650 CT=CT+2:M$=M$+STR$(256*PEEK(CT)+
     PEEK(CT-1)):RETURN
2660 CT=CT+1:M$=M$+STR$(CT-1-(PEEK(CT
     )<128)*(PEEK(CT)+2)-(PEEK(CT)>12
     7)*(PEEK(CT)-254)+DF):RETURN
2670 '
2680 ' * RUTINAS DE VOLCADO DE MEMORI
     A *
2690 '
2700 '       VOLCADO SOBRE PANTALLA
2710 '
2720 '              HEXADECIMAL
2730 FORN=PRGTOFINSTEP10
2740 PRINTUSING"#####";N;:PRINT"  ";
2750 FORC=0TO9:B=PEEK(N+C)
2760 IFB=0THENM$="##"ELSEM$=RIGHT$("0
     "+HEX$(B),2)
2770 IFC=9THENPRINTM$ELSEPRINTM$+" ";
2780 NEXTC,N
2790 END
```

```
2800 '              ASCII
2810 FORN=PRGTOFINSTEP10
2820 PRINTUSING"#####";N;:PRINT"  ";
2830 FORC=0TO9:B=PEEK(N+C)
2840 IFB=0THENC$="##"ELSEC$=" ."
2850 IFB>31THENC$=RIGHT$(" "+CHR$(B),
     2)
2860 IFC=9THENPRINTC$ELSEPRINTC$+" ";
2870 NEXTC,N
2880 END
2890 '          HEX - ASCII
2900 FORN=PRGTOFINSTEP8:CH$=""
2910 PRINTRIGHT$("000"+HEX$(N),4)+"  "
     ;
2920 FORC=0TO7:B=PEEK(N+C)
2930 IFB=0THENM$="##"ELSEM$=RIGHT$("0
     "+HEX$(B),2)
2940 IFB=0THENC$="#"ELSEIFB>31ANDB<12
     8THENC$=CHR$(B)ELSEC$="."
2950 CH$=CH$+C$
2960 PRINT" "+M$;
2970 NEXTC:PRINT"  "+CH$
2980 NEXTN
2990 END
3000 '
3010 '      VOLCADO SOBRE IMPRESORA
3020 '
3030 '              HEXADECIMAL
3040 FORN=PRGTOFINSTEP24
3050 LPRINTUSING"#####";N;:LPRINT"  "
     ;
3060 FORC=0TO23:B=PEEK(N+C)
3070 IFB=0THENM$="##"ELSEM$=RIGHT$("0
     "+HEX$(B),2)
3080 LPRINT" "+M$;
3090 NEXTC:LPRINT:NEXTN
3100 END
3110 '              ASCII
3120 FORN=PRGTOFINSTEP24
3130 LPRINTUSING"#####";N;:LPRINT"  "
     ;
3140 FORC=0TO23:B=PEEK(N+C)
3150 IFB=0THENC$="##"ELSEIFB>31ANDB<1
     28THENC$=RIGHT$(" "+CHR$(B),2)EL
     SEC$=" ."
3160 LPRINT" "+C$;
3170 NEXTC:LPRINT:NEXTN
3180 END
3190 '          HEX - ASCII
3200 FORN=PRGTOFINSTEP16:CH$=""
3210 LPRINTRIGHT$("000"+HEX$(N),4)+"
     ";
3220 FORC=0TO15:B=PEEK(N+C)
3230 IFB=0THENM$="##"ELSEM$=RIGHT$("0
     "+HEX$(B),2)
3240 IFB=0THENC$="#"ELSEIFB>31ANDB<12
     8THENC$=CHR$(B)ELSEC$="."
3250 CH$=CH$+C$
3260 LPRINTM$+" ";
3270 NEXTC:LPRINT"  "+CH$
3280 NEXTN
3290 END
```

El procesador de vídeo del SVI-318/328

Al igual que la pantalla, la tabla de denominaciones se divide en tres partes y cada parte consta de 256 octetos.

El funcionamiento de la tabla en este modo de pantalla es idéntico que en el modo 0, así tenemos que la posición de cada octeto de la tabla se corresponde con la posición de la casilla de pantalla a la que denomina y el número contenido en el octeto indica la posición que ocupa en la tabla de patrones la definición de dicha casilla, pero aquí es donde se encuentra la diferencia en la forma de aplicar estas características.

En el modo cero vimos como el contenido inicial de los octetos de la tabla de denominaciones era un cero, lo que daba a cada casilla de la pantalla la definición del primer carácter de la tabla de patrones, que era el espacio.

Introduciendo un número en uno de los octetos de la tabla de denominaciones en el modo 0 hacíamos aparecer un carácter en la casilla correspondiente de la pantalla, si intentáramos hacer esto en el modo 1 nos llevaríamos un pequeño chasco ya que en la tabla de patrones no está ahora el juego de caracteres.

La tabla de generación de patrones

Al igual que la tabla de denominaciones, esta tabla está dividida en tres partes, veamos en qué lugar de la *VRAM* podemos localizarlas.

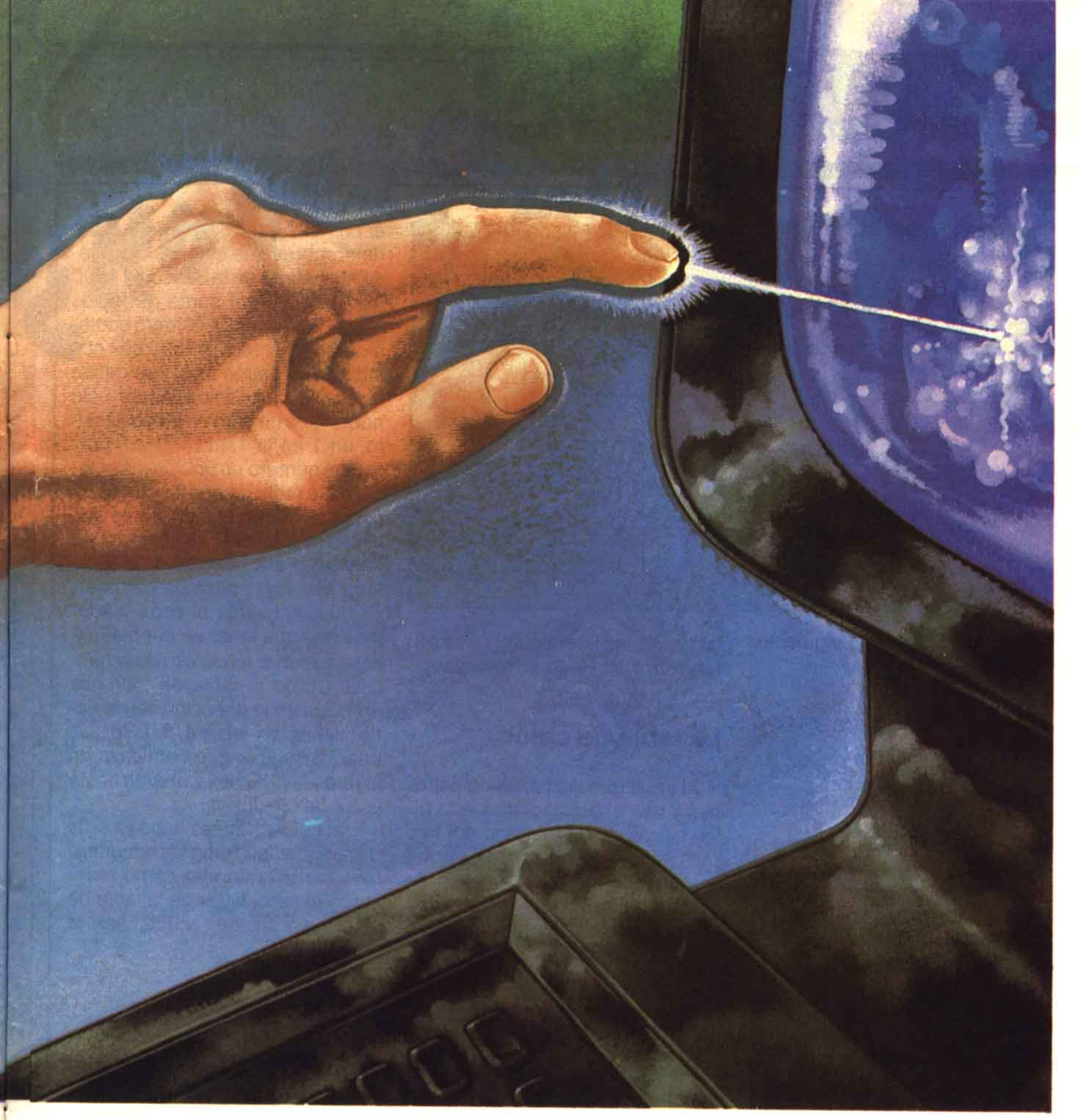
El registro 4 del *VDP* en este modo contiene un 0 que al multiplicarlo por 800 hex nos da como comienzo de la tabla de patrones la posición cero de la memoria de vídeo.

Entre las direcciones cero y 7FF hex de la *VRAM* se definirán las figuras que aparecerán en el tercio superior de la pantalla, entre la 800 hex y la FFF hex se definen las que aparecerán en el tercio medio y entre la 1000 hex y la dirección 17FF hex las del tercio inferior.

A diferencia del modo 0 esta tabla de generación de patrones no contiene inicialmente ninguna definición, o sea, que sus octetos están todos a cero. Cuando nosotros escribimos en este modo de pantalla mediante la instrucción «*PRINT*» el sistema operativo se encarga de trasladar octeto por octeto la definición de las letras desde la zona *ROM* donde se encuentran inicialmente.



Por otra parte el número contenido inicialmente en los octetos de la tabla de denominaciones coincide con la posición que ocupa la casilla correspondiente en la pantalla, con esto se consigue que cualquier modificación que noso-



tros hagamos en las direcciones de la tabla de patrones se verá al momento reflejada en la pantalla de nuestro ordenador.

Si nosotros variamos el contenido de algún octeto de la tabla de denominaciones conseguiremos que en la casilla correspondiente de la pantalla aparezca la definición que inicialmente estaba destinada a otra.

Intentemos clarificar todo esto con un ejemplo:

Primeramente vamos a introducir una figura en el tercio medio de la pantalla...

```
10 SCREEN1:DEFINTA–Z:FOR
   A=0TO7
20 REDD$:CVAL("&B"+D$):
   VPOKE&H850+A,C
30 FORZ=0TO100:NSXT:NEXT
90 GOTO90
100 DATA00111100
110 DATA01000010
120 DATA01100110
130 DATA00011000
140 DATA00011000
150 DATA00111000
160 DATA00011000
170 DATA01111000
```

...tras teclear *RUN* y pulsar *ENTER* aparecerá en la pantalla el gráfico definido mediante el comando *VPOKE* en las direcciones 850 hex a 857 hex de la *VRAM*.

A continuación introducimos otra figura en el tercio superior, pa-

```
FIGURA 6

LA PANTALLA EN EL MODO 1
 ---------------------        ---------------------
| 0 | 1 | 2 | 3 | 4 |<      < 26| 27| 28| 29| 30| 31| Linea 0
|---|---|---|---|---|->      >--|---|---|---|---|---|
|/\/\/\/\/\/\/\/\/ TERCIO ALTO /\/\/\/\/\/\/\/\/\/\/\|
 /\/\/\/\/\/\/\/\/              /\/\/\/\/\/\/\/\/\/\
|224|225|226|227|228|<      <250|251|252|253|254|255| Linea 7
|--------------------->      >----------------------|
| 0 | 1 | 2 | 3 | 4 |<      < 26| 27| 28| 29| 30| 31| Linea 8
|---|---|---|---|---|->      >--|---|---|---|---|---|
|/\/\/\/\/\/\/\/\/ TERCIO MEDIO \/\/\/\/\/\/\/\/\/\/\|
 /\/\/\/\/\/\/\/\/               \/\/\/\/\/\/\/\/\/\
|224|225|226|227|228|<      <250|251|252|253|254|255| Linea 15
|--------------------->      >----------------------|
| 0 | 1 | 2 | 3 | 4 |<      < 26| 27| 28| 29| 30| 31| Linea 16
|---|---|---|---|---|->      >--|---|---|---|---|---|
|/\/\/\/\/\/\/\/\/ TERCIO BAJO  \/\/\/\/\/\/\/\/\/\/\|
 /\/\/\/\/\/\/\/\/               \/\/\/\/\/\/\/\/\/\
|224|225|226|227|228|<      <250|251|252|253|254|255| Linea 23
```

ra ello añade las siguientes líneas...

```
40 FORA=0TO7
50 READD$:C=VAL("&B"+D$):
   VPOKE&H50+A,c
60 FORZ=0TO100:NEXT:NEXT
180 DATA00011000
190 DATA00100100
200 DATA00011000
210 DATA00100100
220 DATA00011000
230 DATA00100100
240 DATA01000010
250 DATA10000001
```

...y ahora una muy sencilla en el tercio inferior:

```
70 VPOKE&H1000,186
```

...finalmente rellenamos cada tercio con la figura definida en una de sus casillas..

```
80 FORA=0TO255:VPOKE&H
   1900+A,VPEEK(&H1900+
   &H50/8):VPOKE&H1800+A,
   VPEEK(&H1800+&H50/8):
   VPOKE&H1A00+A,VPEEK
   (&H1A00):NEXT
```

Mapa de la VRAM en el modo 1

Para realizarlo partimos de las tablas de las figuras 2 y 3 (ver números 14 y 15 de la revista).

La tabla de color

Al igual que las tablas de patrones y de denominaciones esta tabla también está formada por tres partes, y cada octeto de la misma se corresponde con un octeto de la tabla de patrones.

Cada octeto de la tabla define el color que tendrá el fondo y la superficie de un *scan* de 8 *bits* de la pantalla.

En la tabla de la figura siete podemos ver como los cuatro primeros *bits* contienen un número entre 0 y 15 que será el color de fondo, o sea, de los puntos definidos como ceros en el generador de patrones, mientras que los cuatro *bits* finales contienen el color de superficie o unos del generador de patrones.

Con la tabla de color se completa el conjunto de direcciones de la *VRAM* que influyen en la definición de una casilla de la pantalla sin contar con los *sprites*.

Ejemplo: La primera casilla de la pantalla sería la casilla 0 de la figura 6. A esta casilla le corresponde el primer *byte* u octeto de la tabla de nombres que en el modo 1 sería el que está en la dirección 1800 hex de la *VRAM* como podemos ver en la figura 7. El número contenido en la dirección 1800 hex nos indica qué patrón es el que define la casilla cero; inicialmente este número es un cero en la casilla cero, un 1 en la 1, etc.

Si el número en uno de los octetos de la tabla de nombre fuera un 6, por poner un ejemplo, esto significa que la definición de dicha casilla hay que buscarla en el séptimo lugar de la tabla de patrones, ya que se cuenta el cero, y además hay que tener en cuenta que estamos en el tercio alto de la pantalla. Como la tabla de patrones empieza en la dirección cero y cada definición ocupa 8 *bytes*, la nuestra sería la que comienza en la dirección 30 hex y abarca hasta la 37 hex inclusive.

Los colores de los unos y ceros de nuestra definición los encontramos también en el séptimo (contando el cero) lugar de la tabla de color, direcciones 2030 hex a 2037 hex.

Sprites

Cuando empezamos a hablar del procesador de vídeo vimos cómo éste maneja la pantalla mediante la composición de 35 planos superpuestos. 32 de estos planos están exclusivamente dedicados a *sprites* y se numeran de 0 a 31.

Por cada uno de estos planos se moverá un solo *sprite*, que podrá ser:

— Normal a efectos de tablas y de pantalla; definido por 8 octetos en memoria, ocupa 8*8 *píxels* en la pantalla, los *bits* 0 y 1 del registro 1 están a cero.

¿No ves claro tu futuro?

"En los próximos 5 años más del 60 % de las profesiones tendrán relación directa con la informática".

"La preparación que se nece-
hoy es mu

nuevo

curso de INFORMATICA

- LENGUAJES BASIC Y COBOL
- HORARIO OPCIONAL
- MAÑANA, TARDE Y NOCHE
- CURSO DE 12 MESES
- GRUPOS REDUCIDOS
- UN ORDENADOR POR ALUMNO
- ENSEÑANZA INDIVIDUALIZADA
- PRACTICAS PARA EMPRESAS

NOVEDAD: ENSEÑANZA DIRIGIDA POR ORDENADOR

INFORMATE EN:



Enrique Granados, 48, entlo. dcha. - Tel. 253 68 44
BARCELONA
Espoz y Mina, 6 pral. - Tel. 23 16 02-03
ZARAGOZA
Niebla, 15, 1.º, izqda. - Tel. 27 89 71
SEVILLA
Gran Vía, 51, entlo. izqda. - Tel. 25 48 11-12
LOGROÑO

— Normal en las tablas pero aumentado en la pantalla; es definido por 8 octetos de *VRAM*, ocupa 16*16 *píxels*, el *bit* 0 del registro 1 del *VDP* contendrá ahora un 1.

— *Sprite* doble sin aumentar; definido por 16 octetos de *VRAM* y ocupando 16*16 *píxels*, el *bit* 1 del registro 1 está a 1, pero el *bit* 0 contiene un cero.

— *Sprite* doble y aumentado; definido por 16 octetos en la *VRAM*, ocupa 32*32 *píxels* de la pantalla y tanto el *bit* 0 como el *bit* 1 del registro 1 contienen un 1.

No podrán verse más de cuatro *sprites* en una misma línea horizontal, cuando en una línea horizontal coincidan más de cuatro *sprites* se verán sólo aquellos cuatro que se muevan por los planos de menor numeración, se pondrá a uno el *bit* 6 del registro 8 y los *bytes* 0 a 4 de este registro contendrán el número de plano del quinto *sprite*.

Cuando dos *sprites* coincidan en la pantalla se pondrá a 1 el *bit* 5 del registro antes mencionado.

Todo esto ocurre tanto en el modo 1 como en el modo 2 de pantalla, además las tablas que manejan los *sprites* tienen la misma situación en los dos modos de pantalla, por lo tanto todo lo referente a *sprites* se podrá aplicar de igual forma tanto en la pantalla de alta resolución o *SCREEN 1* como en la de baja resolución o *SCREEN 2*.

La tabla de generación de patrones de sprites

En la figura 7 podemos ver que esta tabla comienza en la dirección 3800 hex de la *VRAM* y termina en la 3FFF hex, o sea, al final de la *VRAM*.

Esta tabla la podríamos comparar con el ropero de los *sprites*, ya que en ella se definen las figuras que luego se asignarán a cada *sprite*.

La cantidad de figuras que se pueden definir depende de que utilicemos *sprites* de 8*8, o de 16*16 *píxels*. En el primer caso podremos definir 256 figuras, mientas que en el segundo solamente 64.

En el caso de los *sprites* de 8*8 las definiciones van formándose por grupos de ocho octetos correlativos. Así los ocho primeros octetos de la tabla de patrones definirán la forma del *sprite* que se «vista» con el patrón 0, y que no tiene por qué ser el *sprite* del plano 0 como ya veremos al hablar de la tabla de atributos, los ocho octetos siguientes definirán la forma 1 para *sprites*, etc.

En el caso de *sprites* de 16*16 el orden seguido para la definición es como sigue: los 16 primeros octetos de la definición se refieren a la mitad izquierda de la figura, y los 16 octetos siguientes a la mitad derecha. Los 32 octetos siguientes hacen lo mismo con la siguiente figura para *sprites* y así sucesivamente hasta las 64 figuras definibles.

La tabla de atributos de sprites

Abarca las direcciones 1B00 hex a 1B7F hex de la memoria de vídeo.

En esta tabla hay cuatro *bytes* u octetos por cada uno de los planos de *sprites* que maneja el *VDP* y su significado es el siguiente:

— El primer *byte* indica la coor-

```
FIGURA 7

|---------------------------------------------------------|
|           DISTRIBUCION DE LA VRAM EN EL MODO 1          |
|---------------------------------------------------------|
|DIR. HEX.|                   CONTENIDO                   |
|---------|-----------------------------------------------|
|         |     /TABLA DE GENERACION DE PATRONES          |
|         |                                               |
|0000- 7FF|--Tercio alto.  | Cada 8 octetos forman una fi-|
| 800- FFF|--Tercio medio. | gura a representar en una ca-|
|1000-17FF|--Tercio bajo.  | silla de la pantalla.        |
|---------|-----------------------------------------------|
|         |        /TABLA DE DENOMINACIONES               |
|         |                                               |
|1800-18FF|--Tercio alto.  | Cada octeto corresponde a una|
|1900-19FF|--Tercio medio. | casilla de la pantalla e in--|
|1A00-1AFF|--Tercio bajo.  | dica èl patrón que la define |
|---------|-----------------------------------------------|
|         |     /TABLA DE ATRIBUTOS DE SPRITES            |
|         |                                               |
|  1B00   | Cuatro bytes por cada uno de los 32 planos.   |
|  1B7F   | 2 de coordenadas,1 de tamaño y 1 de color.    |
|---------|-----------------------------------------------|
|1B80-1FFF|                NO UTILIZADO                   |
|---------|-----------------------------------------------|
|         |             /TABLA DE COLOR                   |
|         |                                               |
|2000-27FF|--Tercio alto.  | Color de cada octeto de imagen|
|2800-2FFF|--Tercio medio. |   7  6  5  4  |  3  2  1  0  |
|3000-37FF|--Tercio bajo.  |   Superficie        Fondo     |
|---------|-----------------------------------------------|
|         |  /TABLA DE GENERACION DE PATRONES DE SPRITES  |
|         |                                               |
|  3800   | Se pueden representar 256 figuras de          |
|  3FFF   | 8*8 pixels o 64 figuras de 16*16 pixels.      |
|---------|-----------------------------------------------|
```



denada vertical o «y» del *sprite*.

— El segundo *byte* indica la coordenada horizontal o «x».

— El tercer *byte* indica el «vestido» del *sprite*, o sea, señala cuál de las figuras generadas en la tabla de patrones de *sprites* será la que forma al *sprite*. El número contenido en este *byte* será comprendido entre 0 y 255, pero aumentará de 4 en 4 cuando se traten *sprites* de 16*16.

— El cuarto *byte* tiene dos utilidades:

1) Los *bits* 0 a 3 contienen el color del *sprite*.

2) La longitud horizontal de la pantalla es de 256 *pixels* y resulta que la coordenada horizontal del *sprite*, que se da en el segundo *byte* de los atributos, puede variar entre 0 y 255. Cuando hacemos que un *sprite* desaparezca por la parte izquierda de la pantalla el *byte* que contiene la coordenada x pasa de ser 0 a ser 255 en el momento de la desaparición con lo que tendríamos aún una parte del *sprite* en la izquierda de la pantalla cuando las coordenadas señalan que debería estar ese mismo *sprite* en la derecha de la misma, lo cual es imposible pues un plano de *sprite* sólo puede contener una figura a la vez. Para evitar problemas, cuando hacemos este «mutis» del *sprite* por la izquierda se pone a uno el *bit* 7 del cuarto *byte* de atributos del *sprite* en la tabla de atributos, lo cual indica al *VDP* que el *sprite* no está en el punto 255, sino detrás del cero.

A continuación tenéis un listado de lo que podría ser el boceto inicial de un juego cuya principal característica es que manda datos directamente a la tabla de atributos de *sprites* con los que consigue mover quince de ellos con una velocidad no muy frecuente en el *BASIC*.

El barco lo podéis mover con las teclas del cursor y las teclas definibles 1 y 2 sirven para lanzar cargas de produndidad.

Venerando Solís

```
10 MAXFILES=0:GOSUB390
20 REM****bucle Principal****
30 M=VPEEK(6925):VPOKE6925,M-(M<255)+255*(M=255ANDRND(1)>.88)
40 N=VPEEK(6956):O=VPEEK(6957):VPOKE6957,O-(M+12-O)*(N=200ANDM<239)
50 VPOKE6956,N+(N<200ANDN>95)-105*(N=95)+(191-VPEEK(6924))*(N=200ANDM>0ANDM<238)
60 IFN=95ANDO>BANDO<B+38THENGOTO310
70 P=VPEEK(6988):VPOKE6988,P-(P<192)
80 FORZ=6988TO7000STEP4:IFVPEEK(Z)=106ANDABS(M-VPEEK(Z+1))<10THENVPOKE6925,255:V
POKEZ,192:FORA=0TO10:SOUND7,47:NEXT:SOUND7,255:H=H+50:LOCATE120,0:COLOR5:PRINT"ø
øøø":LOCATE120,0:COLOR1:PRINTH
90 NEXT:M=VPEEK(6929):VPOKE6929,M+(M>0)-255*(M=0ANDRND(1)>.88)
100 N=VPEEK(6960):O=VPEEK(6961):VPOKE6961,O-(M-12-O)*(N=200ANDM>16)
110 VPOKE6960,N+(N<200ANDN>95)-105*(N=95)+(191-VPEEK(6928))*(N=200ANDM>16ANDM<25
5)
120 IFN=95ANDO>BANDO<B+38THENGOTO310
130 P=VPEEK(6992):VPOKE6992,P-(P<192)
140 FORZ=6988TO7000STEP4:IFVPEEK(Z)=122ANDABS(M-VPEEK(Z+1))<10THENVPOKE6929,0:VP
OKEZ,192:FORA=0TO10:SOUND7,47:NEXT:SOUND7,255:H=H+100:LOCATE120,0:COLOR5:PRINT"ø
øøø":LOCATE120,0:COLOR1:PRINTH
150 NEXT:M=VPEEK(6933):VPOKE6933,M-(M<255)+255*(M=255ANDRND(1)>.88)
160 N=VPEEK(6964):O=VPEEK(6965):VPOKE6965,O-(M+12-O)*(N=200ANDM<239)
170 VPOKE6964,N+(N<200ANDN>95)-105*(N=95)+(191-VPEEK(6932))*(N=200ANDM>0ANDM<238
)
180 IFN=95ANDO>BANDO<B+38THENGOTO310
190 P=VPEEK(6996):VPOKE6996,P-(P<192)
200 FORZ=6988TO7000STEP4:IFVPEEK(Z)=138ANDABS(M-VPEEK(Z+1))<10THENVPOKE6933,255:
```

```
VPOKEZ,192:FORA=0TO10:SOUND7,47:NEXT:SOUND7,255:H=H+150:LOCATE120,0:COLOR5:PRINT
"øøøø":LOCATE120,0:COLOR1:PRINTH
210 NEXT:M=VPEEK(6937):VPOKE6937,M+(M>0)-255*(M=0ANDRND(1)>.88)
220 N=VPEEK(6968):O=VPEEK(6969):VPOKE6969,O-(M-12-O)*(N=200ANDM>16)
230 VPOKE6968,N+(N<200ANDN>95)-105*(N=95)+(191-VPEEK(6936))*(N=200ANDM>16ANDM<25
5)
240 IFN=95ANDO>BANDO<B+38THENGOTO310
250 P=VPEEK(7000):VPOKE7000,P-(P<192)
260 FORZ=6988TO7000STEP4:IFVPEEK(Z)=154ANDABS(M-VPEEK(Z+1))<10THENVPOKE6937,0:VP
OKEZ,192:FORA=0TO10:SOUND7,47:NEXT:SOUND7,255:H=H+200:LOCATE120,0:COLOR5:PRINT"ø
øøø":LOCATE120,0:COLOR1:PRINTH
270 NEXT:GOTO30
280 REM***Movimiento del barco***
290 C=STICK(0):B=B+(C=7ANDB>16)-(C=3ANDB<207):VPOKE6913,B:VPOKE6917,B+16:VPOKE69
21,B+32:RETURN
300 REM****Explosion****
310 INTERVALOFF:VPOKE6914,28:VPOKE6918,28:VPOKE6922,28:SOUND7,47:GOTO310
320 NEXT:RETURN
330 REM***lanzamiento cargas profundidad
340 FORA=6988TO7000STEP4:IFVPEEK(A)=192THENVPOKEA,88:VPOKEA+1,B-16:RETURN
350 NEXT:RETURN
360 FORA=6988TO7000STEP4:IFVPEEK(A)=192THENVPOKEA,88:VPOKEA+1,B+45:RETURN
370 NEXT:RETURN
380 REM***inicializacion***
390 SCREEN1,2:DEFINTA-Z:COLOR,5,13:B=120:H=0:SOUND6,1:SOUND9,15:SOUND7,63
400 LINE(0,96)-(255,191),3,BF:LINE(0,0)-(15,191),13,BF:LINE(255,0)-(255,191),13:
CIRCLE(100,30),10,11:PAINT(100,30),11
410 FORA=3TO5STEP2:PUTSPRITEA,(255,(A+3)*16),13,4:PUTSPRITEA+1,(0,(A+4)*16),13,3
:NEXT
420 FORA=11TO14:PUTSPRITEA,(0,200),1,5:NEXT
430 FORA=19TO22:PUTSPRITEA,(0,192),1,6:NEXT
440 REM **Patrones de sprites**
450 FORC=0TO7:FORA=0TO15
460 READD:READE
470 VPOKE&H3800+A+32*C,D:VPOKE&H3810+A+32*C,E
480 NEXT:NEXT
490 PUTSPRITE0,(B,79),1,0:PUTSPRITE1,(B+16,79),1,1:PUTSPRITE2,(B+32,79),1,2
500 ONKEYGOSUB340,360:KEYON
510 ONINTERVAL=15GOSUB290:INTERVALON:RETURN
520 REM**barco**
530 DATA0,0,0,0,0,0,0,0,0,1,0,132,0,99,0,247
540 DATA255,255,127,85,63,255,31,255,15,255,7,255,3,255,1,255
550 DATA1,128,7,192,7,192,69,208,55,228,255,121,170,174,255,255
560 DATA255,255,170,170,255,255,255,255,255,255,255,255,255,255,255,255
570 DATA0,0,0,0,0,0,0,0,0,0,128,3,0,7,128
580 DATA255,255,170,190,255,252,255,248,255,240,255,224,255,192,255,128
590 REM***submarino izquierda***
600 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
610 DATA0,0,8,0,15,0,15,96,127,250,255,255,255,255,127,250
620 REM***submarino derecha***
630 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
640 DATA0,0,0,16,0,240,12,240,95,254,255,255,255,255,95,254
650 REM***torpedo***
660 DATA1,0,3,128,3,128,1,0,2,128,0,0,0,0,0,0
670 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
680 REM***carga de profundidad***
690 DATA3,128,5,192,3,128,0,0,0,0,0,0,0,0,0,0
700 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
710 REM***explosion***
720 DATA130,33,83,34,72,164,38,42,174,72,70,21,34,82,42,100
730 DATA21,164,35,201,131,210,119,244,15,252,135,254,71,170,59,207
```



LA REVISTA IMPRESCINDIBLE
PARA LOS USUARIOS DE LOS ORDENADORES PERSONALES MSX.

Una publicación mensual que ayuda a obtener el máximo partido a su ordenador.

MSX publica cada mes programas y juegos, además de reportajes sobre programación y la posibilidad de ganar premios realizando programas y otros temas siempre de gran interés.

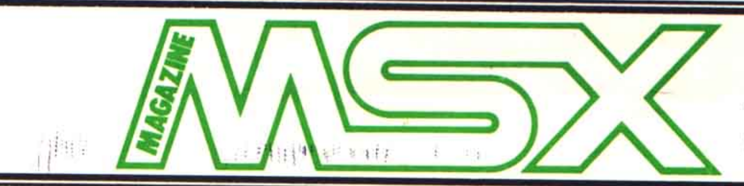




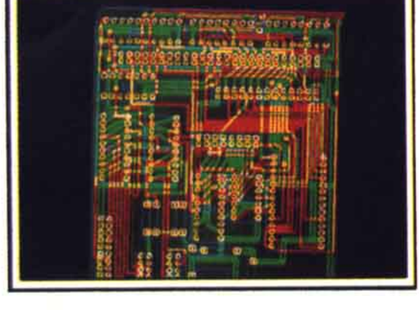
ADEMAS, beneficiese de un **15 % DE DESCUENTO** sobre el precio real de suscripción

PRECIO NORMAL DE SUSCRIPCION	USTED SOLO PAGA	AHORRO
~~3.600~~ PTAS.	3.060 PTAS.	15 %

APROVECHE AHORA esta irrepetible oportunidad para suscribirse a **MSX**. Envie **HOY MISMO** la tarjeta adjunta a la revista, que no necesita sobre ni franqueo. Deposítela en el buzón más cercano. Inmediatamente recibirá su primer ejemplar de **MSX** más el **REGALO**. Y así durante un año (12 números).



Bravo Murillo, 377
Tel. 733 79 69
28020 MADRID



Cómo ahorrar memoria

Es difícil preveer la memoria, pues como ya expusimos la diferente utilidad se presta a todo tipo de conjeturas.

Digamos con ciertos recelos que alrededor de 30 K suponen una buena base, sobre todo si son ampliables. En la actualidad existe cierta tendencia hacia los discos de 3,5 pulgadas que unidos a los de 5,25 son los más representativos en los ordenadores personales a los que nos estamos refiriendo. En cualquier caso para operar con ellos se necesitan de 4 a 8 K aproximadamente, que habrá que restar de la mencionada memoria.

Vamos a exponer un programa modelo con objeto de que al mismo tiempo que seguimos analizando las aplicaciones de la me-

moria de usuario, podamos comprobarlo en pantalla. Nuestra operación no presupone determinados conocimientos de programación.

Modelo A

```
10 DEFINT N-Z
20 DIM N(12)
30 DEF FN ALE(X)=INT(RND(1)
   *6+1)
40 FOR I=1 TO 12
50 N(I)=FN ALE(X)
60 PRINT N(I);
70 NEXT I
80 END
```

Programa de elección de uno entre seis en 12 accesos.

10 Se definen las letras de la 2.ª mitad del alfabeto (26/2) desde la N hasta la Z como enteros, es decir que a cada variable asignada el valor se considera desde el principio un entero. La ventaja es que no es necesario indicar cada vez que se utilicen (INT o %). El principal objetivo es el **ahorro de memoria** cuando no se precisen decimales.

20 Sistema conocido para tomar la previsión que N puede tener hasta doce datos distintos que van:

N(1) ———— N(12)

No sería necesaria esta línea si la cantidad de datos a reservar fuese 10 o menos de 10.

30 Definición de la Función *ALE* como un número tomado aleatoriamente entre seis y entero. En los ordenadores *SVI* para cambiar la cadencia aleatoria debe incrustarse una nueva línea entre la 10 y la 20:

15 X=RND(−TIME)

40 Apertura de un bucle para I doce veces.

50 Asignación de la variable N (recordad que es entero y tiene reserva hasta 12) de tal modo:

N(1)
N(2)
.
= FUNCION DEFINIDA
.
N(12)

Con la función definida *ALE*(X) corresponde a un número del 1 al 6, N(1)... N(12) se convertirán, al ejecutar el programa en números del 1 al 6 pero cada ejecución distintos.

60 Impresión de los citados números uno a continuación de otro (;).

70 Cierre del bucle.

80 Fin.

En este modelo breve de programa se habrá observado que no hay ningún comentario ya que los hemos hecho separadamente, pero los programas suelen guardarse para posteriores ocasiones, ¿quién puede asegurar que se va a recordar de pe a pa para explicárselo a un «colega», por ejemplo, o simplemente para mejorarlo? Para evitar estos programas desasistidos se emplea *REM:*



20 ....: Rem Reserva de lugar para N(12) números tomados al azar.

Con ello hemos aclarado la línea 20, por ejemplo. Es necesario esta aclaración con detalle, ya que afecta a la cantidad de memoria utilizada, aunque no al desarrollo de las instrucciones. Probarlo midiendo la cantidad de memoria antes y después, como ya hemos indicado. Sustituir *REM* por ' y medir de nuevo. Así estableceréis vuestras propias conclusiones.

Modelo B

```
 10 X=127: Y=80
 20 SCREEN 2
 25 OPEN"GRP:" FOR OUTPUT
    AS  1 /// Sólo para MSX
 30 PRESET(10,160)/PRINT
    1,"Actuar G/M/P según radi-
    tos 70/45/25"
 40 A£=INKEY$
 50 IF A$ = "G" THEN R=70 :
    GOTO 90
 60 IF A$ = "P" THEN R = 25 :
    GOTO 90
 70 IF A$ - "M" THEN R = 45 :
    GOTO 90
 80 GOTO 40
 90 CIRCLE (X,Y),R
100 PRESET(10,180):PRINT
    1,"OTRO ? S/N"
120 B$=INKEY$
130 IF BS="s" OR B$="S" THEN
    RUN
140 IF B$="n" OR B$="N" THEN
    END
150 GOTO 120
```

Programa de dibujo para círculos de tres radios.

10 Asignación directa de dos valores fijos, hasta una nueva ejecución.

20 Cambio de sistema de pantalla. Empleamos gran resolución. En los *SVI* será 1 en lugar de 2.

30 Situar en lugar previo lo indicado tras *PRINT*. Es muy importante la estética de las presentaciones que indican la veteranía y el buen hacer, pero además beneficia la claridad aunque se requiera

algo más de memoria.

40 Sistema para suplantar a *INPUT* en alta resolución ya que no es posible hacer uso de él. Se abre un circuito para compulsar si en el teclado hay una de las teclas previstas en acción. Si no el programa se detiene mientras tanto.

50 - 60 - 70 Si se pulsan esas teclas entre comillas se asignarán unos números a R y sigue.

80 Reenvía a línea 40 cuando no hay tecla pulsada.

NOTA: En las líneas 30 y 110 para*SVI* cambiar *PRESET* por *LOCATE.*

90 Se dibuja una circunferencia según el centro indicado en la línea 10, según coordenadas X e Y y uno de los tres radios de 40/80.

100 Relleno de la circunferencia obtenida para formar el círculo, objetivo del programa.

110 Situar en la parte baja de la pantalla, para no superponer en el círculo, una indicación de seguir o dar por terminado el programa principal.

120 Véase línea 40.

130 Si el usuario actúa la letra S mayúscula o minúscula se repetirá la ejecución del programa *(RUN).*
Debe considerarse esta duplicidad de la indicación en la letra S y s como una seguridad para el que utilice el ordenador y no esté muy al tanto de la situación de las mayúsculas o minúsculas, aunque, eso sí, se emplea algo más de *RAM.*

140 Como la línea anterior para terminar con n o N. Estaría mejor emplear en este caso y otros similares un procedimiento que dé fin con cualquier letra y no exclusivamente con la N. Probadlo.

150 Si no hay letra de ninguna tecla actuada vuelve a 120.

Según estos dos ejemplos, si se utilizan programación y memoria según la necesidad de cada proceso se pueden enfocar diferentes tipos de ideas y desarrolarlas.

En las líneas 40/80 hay dos aspectos que debemos comentar:

1 El orden de condiciones no es de mayor a menor porque no hace falta. Sólo se actúa una tecla antes que otra.

2 En la línea 70 se escribe *GOTO* 90 inmediatamente después de los dos puntos (separación de diferentes instrucciones en la misma línea). Se consigue, en este caso ahorrar aunque sólo sea un espacio.

3-4 Para completar este artículo vamos a comentar un nuevo modelo que además será de suma utilidad en adelante.

Como se ha visto prácticamente, hemos analizado dos ejemplos con el principal objetivo de usar convenientemente la *RAM* y si es posible ahorrando *BYTES.*

A continuación vamos a buscar este ahorro por un procedimiento algo más sofisticado.

El sistema binario es tan simple que por ello el ordenador lo entiende a la primera. La acumulación de miles de números 1 y 0 no es problema a un sistema que trabaja a unos 3 o 4 millones de unos y ceros por cada segundo (si, habéis leído correctamente), pero nosotros sufriríamos un empacho cada vez, con ese atracón por muy sencillo que sea. De esto podemos deducir que si queremos que el ordenador imprima:

255 11111111 FF

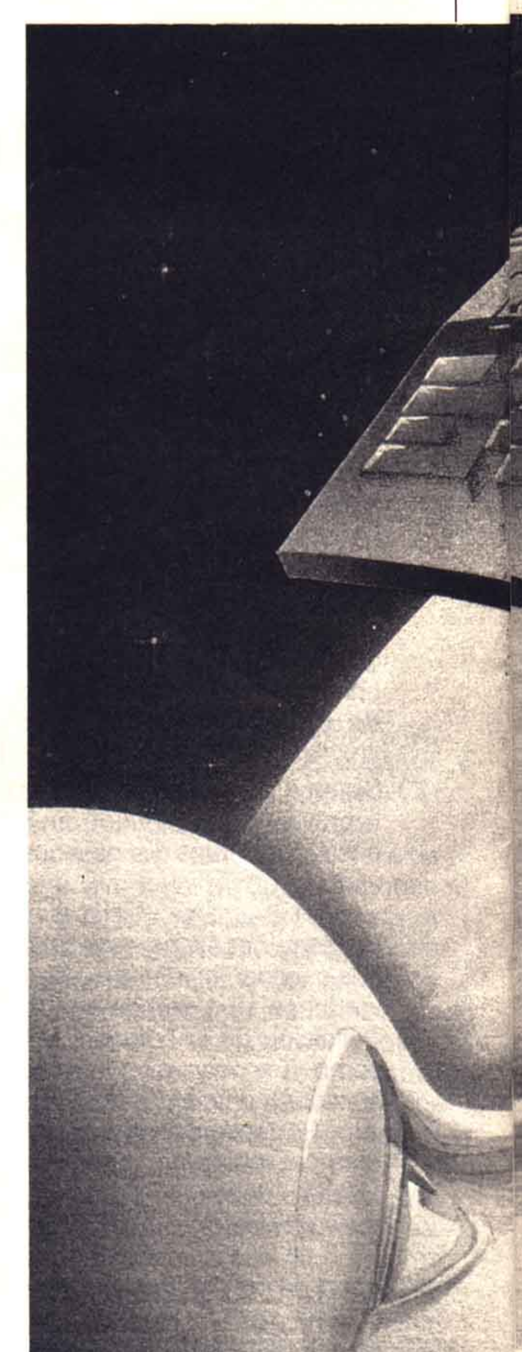
en binario:

PRINT BIN$(255)

o bien:

? &B11111111

Atención a ? que es una fórmula para escribir *PRINT* con mayor co-



modidad.

También podemos escribir el número propuesto en decimal porque el ordenador, mejor dicho su *ULA* (unidad aritmético-lógica) se encarga de intepretarlo adecuadamente:

? 255

o bien en hexadecimal:

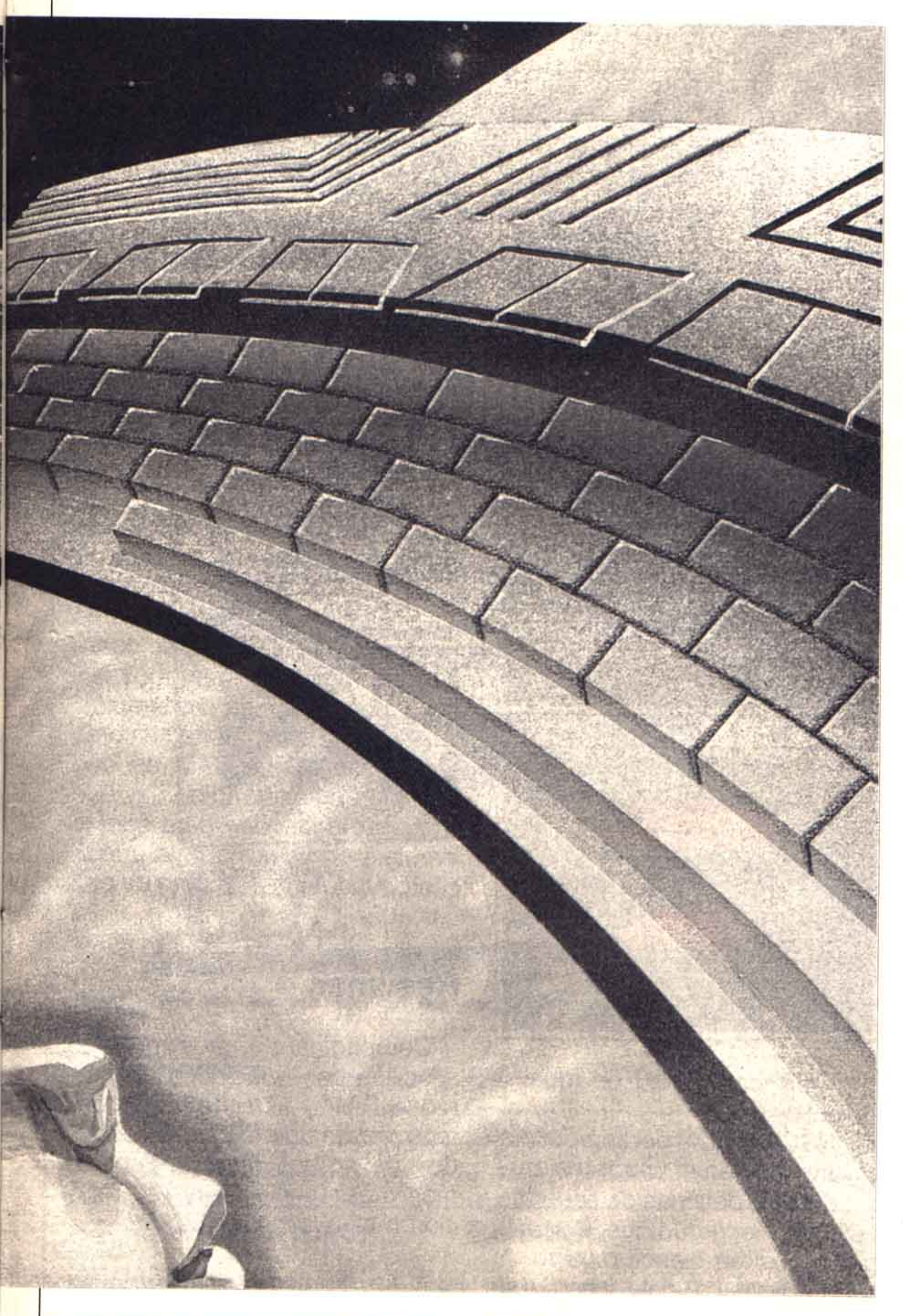
PRINT HEX$(255)

y también:

PRINT &HFF

Surgen inmediatamente las siguientes interrogantes:

¿Cuál de estas formas es la más conveniente para el ordenador? ¿De ellas, cuál será la más rápida? ¿Con qué fórmula emplearemos menos memoria?



Combinando las tres respuestas que demos, las aplicaré, según mis necesidades.

Vayamos al primer modo (255):

Para nosotros es el más comprensible y no ocupa demasiada memoria. Sin embargo, es poco manejable en cadenas y la máquina necesita tiempo para su interpretación, ya que es el más alejado del circuito lógico.

El segundo modo (11111111):

Aunque tenga el mismo valor que 255 en decimal su «acogida» por la máquina es inmediata y por ello más rápido su tratamiento, pero ya se ve que ocupa más del doble de memoria en *BASIC* y su uso es farragoso y lento para nosotros. Propende al error y por ello lo que ganamos en velocidad por un lado lo perdemos en cuanto a su manejo por otro.

La utilidad del modo binario en *SPRITES* es la máa clara y rápida. Veremos su aplicación y ventajas con vuestra benevolencia en próximos artículos.

Y el tercer modo (FF):

Siendo la aplicación hexadecimal del número propuesto permite adaptarse a un punto intermedio entre la máquina y el usuario, por ello se utiliza en el lenguaje previo al código máquina y sus listados no se prestan a confusión como los escritos en binario. Estamos en el centro: comprensión media y ocupación mínima de memoria para números medios y grandes:

Decimal	Hexadecimal	Binario
65535	ff	1111111111111111

Realizad un breve programa incluyendo estos números en cada ocasión. Probad la memoria utilizada y almacenad vuestras anotación más anotación.

Modelo C

Con objeto de completar lo expuesto en 4.3 sugerimos el presente modelo:

```
10 INPUT"Número decimal"; NC
20 Print"Hexadecimal de";NC;
   "–";HEX$(NC)
30 INPUT"Número decimal"; NB
40 Print"Binario           de";NB;"–
   ";BIN$(NB)
50 INPUT" OTRA VEZ S/N"; R$
60 IF R$="S" OR R$="s" THEN
   RUN
70 IF RS="N" OR R$="n" THEN
   END
```

Como comentario general este modelo se utiliza para obtener el hexadecimal y el binario de un número decimal.

O bien se pueden obtener directamente cuando no se precisen más que unos pocos y con &H, &B y a continuación el número deseado cuando se prefiera el decimal.

Modelo D

La velocidad de ejecución no siempre está ligada a la ocupación de memoria, pero en muchas ocasiones es necesario sacrificar la segunda en aras al incremento de la primera:

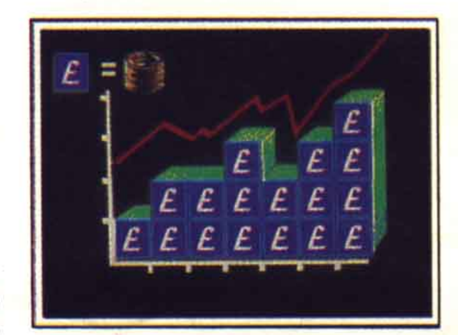
```
10 DEFINT Q-Z
20 TIME=0
30 FOR A=0 TO 12000: NEXT A
40 T=TIME
50 Pi=3.14: CA=15.9; CN=Pi*CA:
   CP=CN/10
60 PRINT T/CP;"      Décimas de
   segundo."
70 INPUT"Otra vez S/N; M$
80 If M$="S" THEN RUN ELSE
   END
```

10 Recordad la misma línea de 4.1.

20 Asignar el reloj del sistema al inicio, es decir para que empiece desde 0.

30 Bucle de tiempo. Estos bucles se emplean para ralentizar secciones de programas que lo precisen. En este caso servirá como base de cálculo de tiempo.

40 Cálculo de tiempo parcial tomado en décimas de segundo. Para otra relación (minutos, segundos, etc.) variar la relación:





CP=cn/10 (CN es igual a cn)

50 INSCRIPCION del tiempo total invertido en el programa, medido en décimas de segundo.

NOTA: Recordad que ▲ significa espacio en blanco o vacío.

60/70 Repetir o finalizar según se elija. Si salen diferentes valores hallar la media entre éllos.

Tomad buena nota del tiempo o de la medida que os haya salido y a continuación cambiar la línea 30:

```
30 FOR W=0 TO 12000: NEXT
```

Ejecutad el modelo con esta variante y anotad el nuevo tiempo. Aunque la reducción no sea muy grande será casi una tercera parte del tiempo inicial antes de modificar la línea 30. Esto se debe a dos razones:

La primera se esbozó en 3.2 Exponíamos que la precisión de un entero (línea 20) al ser inferior que un número corriente usaba menos *RAM* y por ello menos tiempo ya que puede emplear mayor velocidad.

La segunda es debida a la supresión de la información de la variable después de *NEXT* (T).

Sin embargo, esta forma de componer programas provoca no pocas confusiones sobre todo en los bucles concatenados y por lo tanto sólo deberá utilizarse cuando los programas sean muy largos y con la debida preparación. El tiempo es tan importante como el espacio libre de *RAM*, así que en programación rápida hay que sacrificar la claridad para que se aventaje en velocidad, ya que el código máquina es harina de otro costal.

Resumen

Compaginando unos modelos sencillos con relaciones de memoria *RAM* y tiempos reales hemos presentado las particularidades de ambos medios. Esperamos vuestras iniciativas y en relación a ellas seguiremos.

José Leal Rodríguez

CURSO DE INGLES

The Gruneberg Linkword Language System es un sistema, para enseñanza de idiomas, más rápido y fácil que los métodos convencionales aplicados actualmente.
En poco tiempo, máximo 20 horas, te enseñará un vocabulario de 400 palabras y adquirirás unas buenas nociones de gramática. Esto te permitirá entender y ser entendido en tus viajes a lugares de habla inglesa o en tus contactos con personas que se expresen en ese idioma.
Por otra parte, el Sistema PlusData, consigue que el ordenador se convierta en un perfecto profesor que te explicará, orientará y corregirá, manteniendo en todo momento un "diálogo" interactivo de resultados sorprendentes.



–L. Taylor. **"POPULAR COMPUTER WORLD":**
"Quedé francamente atónito al comprobar la efectividad de la sugestión de imágenes como elemento de ayuda a la retención..."

–**"PERSONAL COMPUTER WORLD":**
"Un suceso fuera de serie..."

–Bill Barnet. **"COMPUTER CHOICE":**
"De todos los paquetes para aprender idiomas éste es el más interesante..."



Programas de EAO para EGB.
Cursos de Basic, Cobol, etc. AUTODIDACTAS.

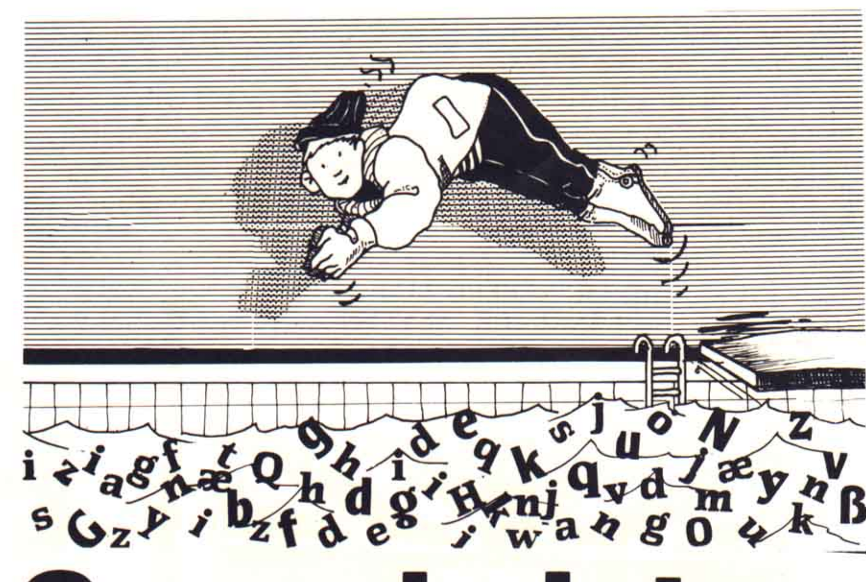
Nombre ....
Apellidos ....
Dirección ....
Población ....
D.P. .... Tlno. ....
Forma de pago: Reembolso ☐ Giro postal ☐ Envío talón ☐

☐ Curso de Inglés 1.ª parte. 10 lecciones Linkword. (Cinta) P.V.P. 6.900.-Ptas.
☐ Curso de Inglés 1.ª parte. 10 lecciones Linkword. (3,5"-Disk) P.V.P. 7.900.-Ptas.
ENVIAR ESTE CUPON A: PLUS DATA, S.A. C/. GRAN VIA, 661 pral. 08010-Barcelona. Tel. 246 02 02

La abundancia de pasatiempos ha llevado a la creación de revistas especializadas en ellos. Todos habremos hecho, alguna que otra vez, un crucigrama, una sopa de letras, un jeroglífico, etc. Sin embargo, estos pasatiempos se han ido desarrollando en los ordenadores, pudiendo obtener múltiples combinaciones de un mismo juego.

En esta sopa de letras tenemos que destacar, que la complejidad puede correr a cargo de cada uno, pues el bloque formado por las líneas 400 a 490, son las que cargan los datos que habrá que descubrir.

Williams Hernández
Santa Cruz de Tenerife



Sopa de letras

```
10 '❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋
20 '❋❋❋❋❋ Programa elaborado por ❋❋❋❋
30 '❋❋❋❋❋    Williams Hernández    ❋❋❋❋
40 '❋❋❋❋❋        ♥ TENERIFE 85     ❋❋❋❋
50 '❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋❋
60 '================================
70 '========== PRESENTACION ==========
80 '================================
90 SCREEN1:KEYOFF:WIDTH 32:CLEAR700:COLOR 11,13
100 LOCATE 9,0:PRINT"SOPA DE LETRAS"
110 LOCATE 9,1:PRINT"──────────────"
120 LOCATE 2,22:PRINT"Pulsa una tecla para empezar"
130 X=RND(1)*32:Y=3+RND(1)*17:Z=65+RND(1)*25
140 LOCATE X,Y:PRINTCHR$(Z)
150 IF INKEY$="" THEN 130
160 CLS
170 LOCATE 10,1:PRINT"SOPA DE LETRAS"
180 LOCATE 10,2:PRINT"──────────────"
190 LOCATE 5,3:PRINT"Este programa consiste en"
200 LOCATE 2,4:PRINT"encontrar los nombres de 12"
210 LOCATE 2,5:PRINT"países americanos, los cuales"
220 LOCATE 2,6:PRINT"se hallan ocultos en una SOPA"
230 LOCATE 2,7:PRINT"DE LETRAS."
240 LOCATE 5,9:PRINT"Deberás dar el nombre del"
250 LOCATE 2,10:PRINT"país y las coordenadas de la"
260 LOCATE 2,11:PRINT"primera letra; primero la fila"
270 LOCATE 2,12:PRINT"y después la columna."
280 LOCATE 5,14:PRINT"Por ejemplo:"
290 LOCATE 2,16:PRINT"Palabra ?  NICARAGUA"
300 LOCATE 2,17:PRINT"Coordenadas(f,c) ?  3,14"
310 LOCATE 2,21:PRINT"Pulsa una tecla para continuar"
320 IF INKEY$="" THEN 320
330 CLS
340 LOCATE 9,0:PRINT"SOPA DE LETRAS"
350 LOCATE 9,1:PRINT"──────────────"
360 LOCATE 5,5:PRINT"Si los datos son correctos,  oirás una fanfarria y el nom-   bre del país se imprimirá a la  izquierda de la pantalla."
370 LOCATE 5,10:PRINT"Las palabras están escritas  en horizontal (hacia la dere-   cha),en vertical (hacia abajo) y en diagonal (a la derecha y   hacia abajo)."
380 LOCATE 2,18:PRINT"Pulsa una tecla para continuar"
390 IF INKEY$="" THEN 390
400 '==============================
410 '======= CARGA DE DATOS =======
420 '==============================
430 DIM PALABRA$(11,2)
440 FOR I=0 TO 11
450 READ PALABRA$(I,0)
460 NEXT I
470 DATA GUATEMALA,NICARAGUA,MEXICO,COLOMBIA,BOLIVIA,VENEZUELA,BRASIL,ECUADOR,PANAMA,CANADA,CHILE,PERU
480 CLS
```

```
490 LOCATE 3,12:PRINT"Espere un momen
    to, por favor"
500 '===============================
510 '===== COLOCACION DE DATOS =====
520 '===============================
530 DIM TABLA$(14,14)
540 '======== HORIZONTALES ========
550 FOR I=0 TO 3
560 X=INT(RND(1)*14):Y=INT(RND(1)*14)
570 IF X+LEN(PALABRA$(I,0))>14 THEN 5
    60
580 '====== COLOCACION PRUEBA ======
590 FOR M=1 TO LEN(PALABRA$(I,0))
600 IF TABLA$(X+M,Y)="" OR TABLA$(X+M
    ,Y)=MID$(PALABRA$(I,0),M,1) THEN
     NEXT M ELSE 560
610 '===== COLOCACION PALABRA$ =====
620 PALABRA$(I,1)=STR$(X+2):PALABRA$(
    I,2)=STR$(Y+1)
630 FOR M=1 TO LEN(PALABRA$(I,0))
640 TABLA$(X+M,Y)=MID$(PALABRA$(I,0),
    M,1)
650 NEXT M
660 NEXT I
670 '========= VERTICALES =========
680 W=0
690 FOR I=4 TO 7
700 X=INT(RND(1)*14):Y=INT(RND(1)*14)
710 IF Y+LEN(PALABRA$(I,0))>14 THEN 7
    00
720 W=W+1:IF W=50 THEN ERASE TABLA$:G
    OTO 530
730 '====== COLOCACION PRUEBA ======
740 FOR M=1 TO LEN(PALABRA$(I,0))
750 IF TABLA$(X,Y+M)="" OR TABLA$(X,Y
    +M)=MID$(PALABRA$(I,0),M,1) THEN
     NEXT M ELSE 700
760 '===== COLOCACION PALABRA$ =====
770 PALABRA$(I,1)=STR$(X+1):PALABRA$(
    I,2)=STR$(Y+2)
780 FOR M=1 TO LEN(PALABRA$(I,0))
790 TABLA$(X,Y+M)=MID$(PALABRA$(I,0),
    M,1)
800 NEXT M
810 NEXT I
820 '========= DIAGONALES =========
830 W=0
840 FOR I=8 TO 11
850 X=INT(RND(1)*14):Y=INT(RND(1)*14)
860 IF X+LEN(PALABRA$(I,0))>14 OR Y+L
    EN(PALABRA$(I,0))>14 THEN 850
870 W=W+1:IF W=50 THEN ERASE TABLA$:G
    OTO 530
880 '====== COLOCACION PRUEBA ======
890 FOR M=1 TO LEN(PALABRA$(I,0))
900 IF TABLA$(X+M,Y+M)="" OR TABLA$(X
    +M,Y+M)=MID$(PALABRA$(I,0),M,1)
    THEN NEXT M ELSE 850
910 '===== COLOCACION PALABRA$ =====
920 PALABRA$(I,1)=STR$(X+2):PALABRA$(
    I,2)=STR$(Y+2)
930 FOR M=1 TO LEN(PALABRA$(I,0))
940 TABLA$(X+M,Y+M)=MID$(PALABRA$(I,0
    ),M,1)
950 NEXT M
960 NEXT I
970 '===============================
980 '======= SOPA DE LETRAS ========
990 '===============================
1000 FOR I=0 TO 14
1010 FOR J=0 TO 14
1020 Z=65+RND(1)*25
1030 IF TABLA$(I,J)="" THEN TABLA$(I,
     J)=CHR$(Z)
1040 NEXT J,I
1050 '===============================
1060 '====== COLOCACION TABLA$ ======
1070 '===============================
1080 CLS:COLOR 15,4
1090 LOCATE 23,0:PRINT"111111"
1100 LOCATE 14,1:PRINT"12345678901234
     5"
1110 LOCATE 14,2:PRINT"▲▲▲▲▲▲▲▲▲▲▲▲▲▲
     ▲"
1120 FOR I=1 TO 9
1130 LOCATE 12,2+I:PRINTCHR$(48+I);CH
     R$(208);SPC(15);CHR$(207);CHR$(4
     8+I)
1140 NEXT I
1150 FOR I=10 TO 15
1160 LOCATE 11,2+I:PRINTCHR$(49);CHR$
     (38+I);CHR$(208);SPC(15);CHR$(20
     7);CHR$(49);CHR$(38+I)
1170 NEXT I
1180 LOCATE 14,18:PRINT"▼▼▼▼▼▼▼▼▼▼▼▼▼
     ▼▼"
1190 LOCATE 23,20:PRINT"012345"
1200 LOCATE 14,19:PRINT"1234567891111
     11"
1210 FOR I=0 TO 14
1220 FOR J=0 TO 14
1230 LOCATEI+14,J+3:PRINTTABLA$(I,J)
1240 NEXT J,I
1250 LOCATE 3,0:PRINT"● Williams"
1260 '===============================
1270 '========= PREGUNTAS =========
1280 '===============================
1290 LOCATE 2,21:PRINTSPC(25)
1300 LOCATE 2,22:PRINTSPC(25)
1310 LOCATE 2,21:INPUT"Palabra";RE$
1320 LOCATE 2,22:INPUT"Coordenadas(f,
     c)";A$,B$
1330 FOR I=0 TO 11
1340 IF RE$=PALABRA$(I,0) THEN IF " "
     +A$=PALABRA$(I,2) AND " "+B$=PAL
     ABRA$(I,1) THEN LOCATE 2,I+1:PRI
     NTPALABRA$(I,0):PLAY"V15O6L8CDEF
     G":N=N+1:IF N=12 THEN 1380 ELSE
     1290
1350 NEXT I
1360 PLAY "V15O1C8D4E2"
1370 GOTO 1290
1380 LOCATE 2,21:PRINTSPC(25)
1390 LOCATE 2,22:PRINTSPC(25)
1400 LOCATE 10,21:PRINT"Otra vez (S/N
     )?"
1410 IF INKEY$="S" THEN ERASE TABLA$:
     GOTO 480
1420 IF INKEY$="N" THEN END ELSE 1410
```

Código máquina

Por fin... por fin terminamos la parte teórica pura de este cursillo. Hoy abordaremos las instrucciones de intercambio, transferencia y búsqueda de bloques, y las de entrada y salida. En el próximo número tendréis un desensamblador que nos ha enviado un lector, y que os permitirá examinar programas que hayáis tecleado de libros o revistas, así como hechar un vistazo a la ROM de vuestro MSX.

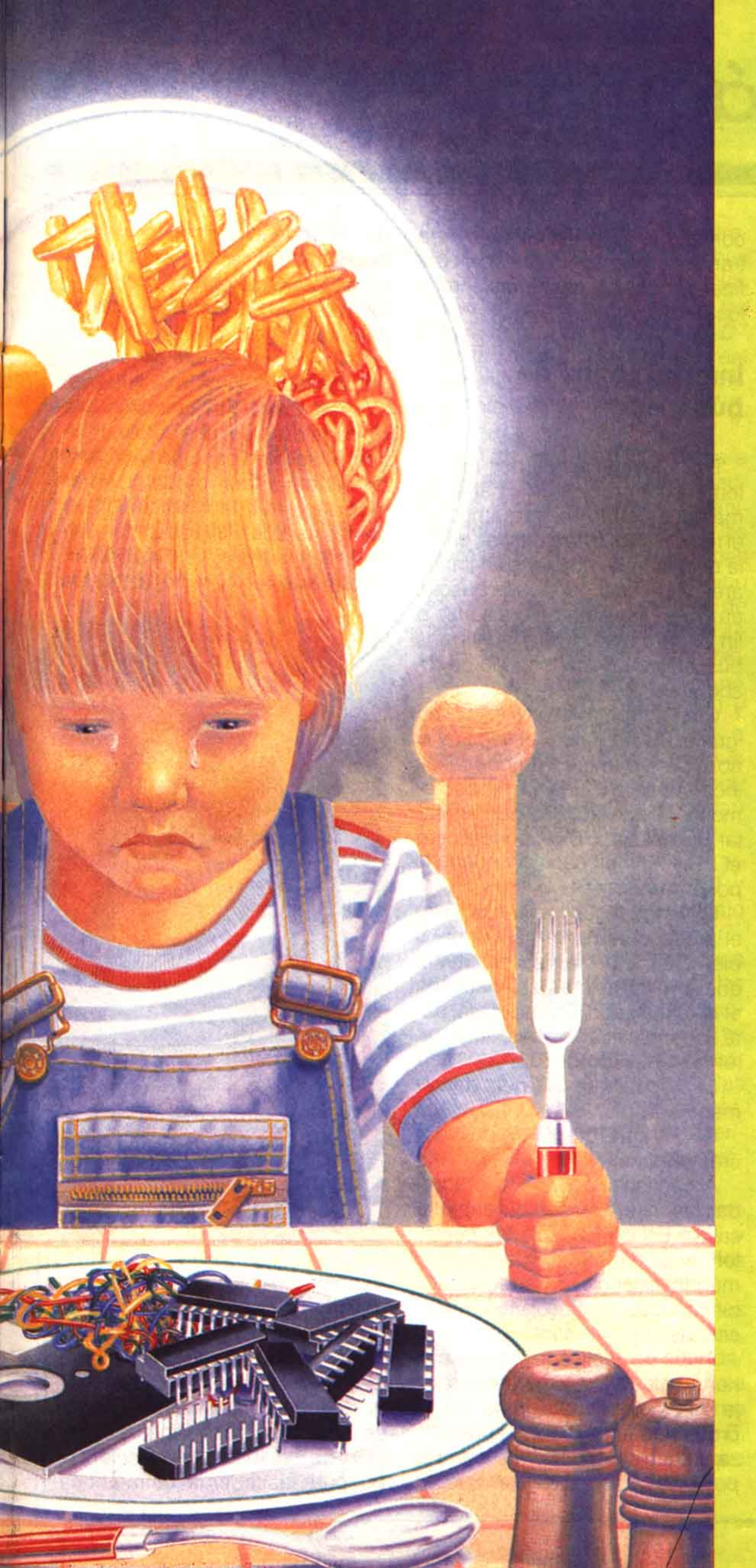
Vamos sin más con las instrucciones de intercambio.

EX DE,HL: Esta instrucción intercambia el contenido de estos dos registros dobles. Por ejemplo, si *HL* contiene 5003H y *DE* contiene *AF00H,* después de ejecutar *EX DE,HL HL* contendrá *AF00H* y *DE 5003H.* Esta instrucción no afecta a ningún *flag.*

EX AF,AF': El microprocesador *Z80* posee dos juegos idénticos de registros, y al segundo se le distingue poniéndole al nombre del registro la comilla simple ('). Así, el primer juego de registros lo forman A, B, C, D, E, H, L, F y el segundo A', B', C', D', E', H', L' y F'. Sin embargo, no podemos pasar de uno a otro de forma sencilla, sino únicamente a través de esta instrucción y la siguiente. De este modo *EX AF,AF'* hace que el contenido actual del acumulador y los *flags* pase «a segundo plano» pasen a primer plano. Evidentemente, con una segunda instrucción *EX AF,AF'* recuperamos los contenidos que teníamos al principio, por lo cual esta instrucción puede ser muy útil para preservar el estado del microprocesador (los *flags*) después de ejecutar una serie de instrucciones.

Es importante destacar que no





existen operaciones tales como *LD A',3FH,* sino que seguimos utilizando las instrucciones habituales, sólo que ahora el registro «de primer plano» contiene lo que tenía antes el de «segundo plano».

Esta instrucción no afecta a ningún *flag.*

EXX: Esta instrucción realiza simultáneamente tres operaciones que no existen individualmente, y que podríamos representar por *EX BC,BC', EX DE,DE', EX HL,HL'.* Lo que conseguimos es preservar el contenido de todos los demás registros que tienen un juego alterno (observad que *IX* e *IY* son únicos, por lo cual no hay operación *EXX* para ellos). Tampoco afecta a los *flags.*

EX (SP),HL: Recordaréis que el *Z80* maneja una pila o *stack* para guardar las direcciónes de retorno de las llamadas a subrutina y los datos que podemos guardar allí con *PUSH* y recuperar con *POP.* La dirección de la cual se recuperan datos se conserva en el registro *SP,* y se representa por *(SP).* En general, un registro entre paréntesis significa contenido de la dirección a la que apunta dicho registro. Por tanto, la instrucción *EX (SP),HL* recupera un dato de la pila SIN CAMBIAR EL PUNTERO DE LA PILA *SP,* y además sustituye al dato recuperado con el contenido de *HL.* Es decir, no cambia el contenido de *HL* con el contenido de *SP,* sino con el contenido de la dirección a la que apunta *SP.* Por supuesto, como *HL* es un registro doble, el intercambio es de dos *bytes:* el contenido en L pasa a *(SP),* y el contenido en H pasa a *(SP+1).*

Tampoco afecta a ningún *flag.*

EX (SP),IX: Exactamente igual que la anterior, pero con *IX* en lugar de *HL.*

EX (SP),IY: Exactamente igual

que la anterior, pero con *IY* en lugar de *HL*.

Instrucciones de transferencia de bloques.

Estas instrucciones funcionan todas de forma similar. Antes de ejecutarlas hemos de introducir en *HL* la dirección de comienzo o final del bloque a transferir, según la operación. En *DE* hemos de introducir la dirección de comienzo o final del área de memoria al cual queremos transferir los datos. Y en *BC* hemos de introducir el número de *bytes* a transferir. Este registro se utiliza como contador para realizar un bucle.

LDI: Transfiere un *BYTE* de la dirección a la que apunta *HL* a la dirección a la que apunta *DE*, esto es, *(HL)—>(DE)*. El contenido anterior de *(DE)* se pierde. Inmediatamente y de forma automática *HL* y *DE* se incrementan en uno y *BC* se decrementa en uno, terminando la instrucción. Los *flags* H y N se ponen a cero, y si el resultado de decrementar *BC* es que *BC=0*, entonces el *flag P/V* se pone a cero, poniéndose a uno si *BC<>0*. Esto es muy útil para detectar el final de la transferencia.

LDIR: Esta instrucción realiza la misma transferencia que la anterior, incrementa igual *HL* y *DE* y decrementa *BC*, y se repite sin cesar hasta que *BC=0*. Esto permite mover un bloque de memoria de cualquier tamaño con una sola instrucción. Puesto que los punteros *HL* y *DE* se incrementan, deben cargarse con la dirección de comienzo del bloque a transferir y la dirección de destino del área al cual se transfiere.

LDD: Esta es igual que *LDI*, pero en lugar de incrementar *HL* y *DE*, los decrementa.

LDDR: Igual que *LDIR*, pero decrementando los punteros *HL* y *DE*. Por tanto, *HL* deberemos cargarlo con la dirección del último *byte* a transferir, y *DE* con la dirección del final del área al cual transferimos los datos.

Instrucciones de búsqueda

Estas instrucciones nos permiten buscar un *byte* en un área de memoria. Para ello, cargamos en el registro A el *byte* a buscar, en *HL* la dirección de comienzo o fin del área en el cual queremos buscar, en *HL* la dirección de comienzo o fin del área en el cual queremos buscar, y en *BC* la longitud de dicho área.

CPI: Compara A con *(HL)*. Si son iguales, pone a cero el *flag* Z, y si no lo pone a uno. En ambos casos incrementa en uno *HL* y decrementa en uno *BC*. Si al decrementar *BC* obtiene *BC=0*, pone a cero el *flag P/V*, y en caso coontrario lo pone a uno.

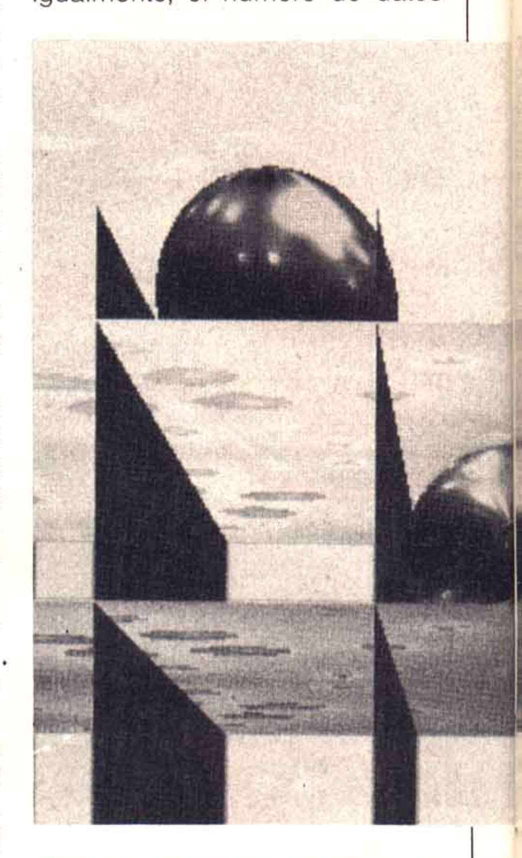
CPIR: Igual que la anterior, pero el proceso se repite hasta que, o bien *BC=0*, o bien se encuentra que *A=(HL)*, esto es, se encuentra el *byte* buscado. En este caso, para saber su situación basta con mirar el contenido de *HL*.

CPD: Igual que *CPI*, pero decrementando *HL*.

CPDR: Igual que *CPIR*, pero decrementando *HL*.

A continuación vamos a abordar las instrucciones en entrada/salida. Estas instrucciones permiten al *Z80* comunicarse con el mundo exterior, enviando y recibiendo información en paralelo, esto es, un *byte* cada vez.

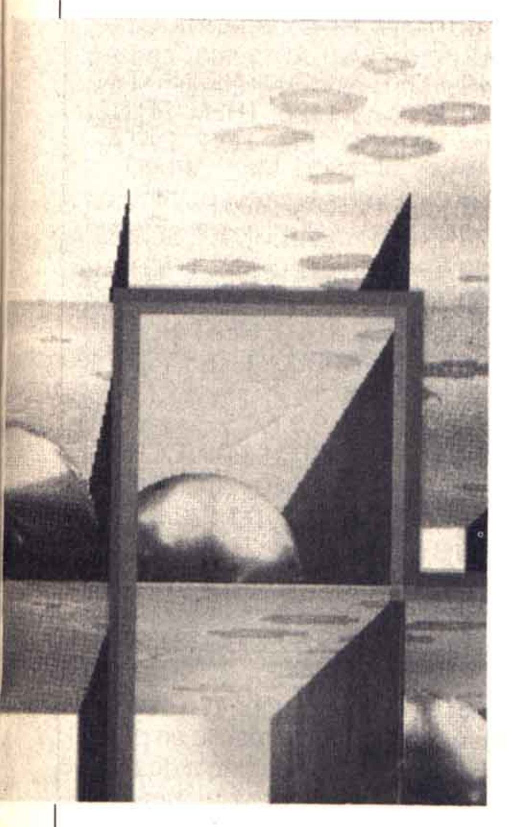
Antes de explicar las instrucciones, vamos a explicar cómo trabaja el *Z80* en entrada/salida. Como probablemente ya sabréis, se utiliza un bus de direcciones formado por dieciséis pistas y un bus de datos formado por ocho pistas. Al decir bus nos referimos a un grupo de conductores eléctricos por los que circula simultáneamente información, de modo que al juntar la información de cada línea del bus obtenemos una información completa. Por tanto, los *bits* de cada pista del bus de direcciones, juntos, indican una dirección de memoria, por lo cual, tendremos que el número total de direcciones es dos elevado al número de pistas del bus, esto es 65536. Igualmente, el número de datos



distintos posibles que se pueden enviar por el bus de datos es de dos elevado a ocho, esto es 256. Sin embargo, cuando el *Z80* está ejecutando una instrucción de entrada/salida, el bus de direcciones se utiliza para decodificar la dirección de un dispositivo periférico, no para acceder a la memoria

central. Para que tanto los periféricos como la memoria sepan a qué atenerse, se utiliza una señal de control denominada *IORQ* (*Input/ Outpur Request* = Solucitud de entrada salida), que se activa o no según el *Z80* ejecute una instrucción de entrada/salida o no.

Al usar el bus de direcciones para decodificar periféricos, esto significa que el *Z80* puede, en principio, enviar datos a uno de entre 65536 periféricos distintos. Sin embargo, normalmente no se utilizan tantos, por lo cual cada fabricante de ordenador suele decidirse por usar o bien los ocho *bits* bajos o los ocho *bits* altos del bus de direcciones, lo que en cualquier caso pisibilita direccionar 256 periféricos, lo cual es más que suficiente.



Por supuesto, el dato a enviar o recibir circula por el bus de datos. Al explicar las instrucciones representamos las líneas 0 a 7 del bus de direcciones por *A0-A7*, las líneas 8 a 15 por *A8-A15*, las líneas del bus de datos por *D0-D7* y los registros por su nombre real o por el nombre genérico r, que representará a los registros A, B, C, D, E, H o L.

IN A,(n): El operando n pasa a las líneas 0-7 del bus de direcciones *(n—>A0-A7)*, el contenido del acumulador pasa a las líneas 8-15 del bus de direcciones *(A—>A8-A15)*, y el dato recibido del periférico a través del bus de datos pasa al acumulador *(D0-D7—>A)*. No afecta a ningún *flag*.

IN r,(C): La resumimos como *(C—>A0-A7), (B—>A8-A15)* y *(D0-D7—>r)*. El *flag* N queda a cero, los *flags* Z, S y H quedan afectados según sea el dato recibido y el *flag* P/V indica la paridad del dato recibido.

OUT (n),A: La resumimos como *(n—>A0-A7), (A—>A8-A15)* y *(A—>D0-D7)*. Como veis, el dato a enviar ha de ser introducido previamente en el acumulador, cuyo contenido aparece también en las líneas 8-15 del bus de direcciones, por lo cuál esta instrucción se usa normalmente para decodificar periféricos que utilicen sólo las líneas 0-7 del bus de direcciones. En este caso la dirección del periférico será el operando n. No afecta a ningún *flag*.

OUT (C),r: La resumimos como *(C—>A0-A7), (B—>A8-A15)* y *(r—>D0-D7)*. No afecta a los *flags*.

Al igual que en el caso de las instrucciones de transferencia, existen instrucciones de entrada/ salida de bloque. En éstas no se utiliza el registro doble *BC* como contador, sino el registro simple B, por lo cual el número de datos que manejaremos de una sola vez será de, como máximo, 256. Se usa el registro *HL* como puntero, y antes de ejecutar la instrucción deberemos cargarlo con la dirección de comienzo o final del bloque de datos a enviar, o del área de memoria donde queremos almacenar los datos recibidos.

INI: En primer lugar se decodifica la dirección del periférico *(C—>A0-A7), (B—>A8-A15)*. A continuación el dato recibido se envía a la dirección de memoria a la que apunta *HL (D0-D7—>(HL))*. Por último, se incrementa en uno *HL (HL=HL+1)* y se decrementa *B (B=B−1)*. Si al decrementar B nos da cero, el *flag* Z se pone a uno. El *flag* N queda siempre a uno.

INIR: Funciona exactamente igual que *INI*, pero tras decrementar B se vuelve a empezar, deteniéndose el proceso cuando B vale cero. Los *flags* Z y N quedan a uno.

IND: Es igual que *INI*, pero el registro *HL* se decrementa en lugar de incrementarse *(HL=HL−1)*. En este caso, antes de ejecutar la instrucción, cargaremos *HL* con la dirección final del área donde queremos almacenar los datos.

INDR: Es igual que *INIR*, pero el registro *HL* se decrementa en lugar de incrementarse *(HL=HL−1)*.

OUTI: La representamos como *(C—>A0-A7), (B—>A8-A15), (r—>D0-D7), (HL=HL+1), (B=B−1)*. Los *flags* quedan igual que en *INI*.

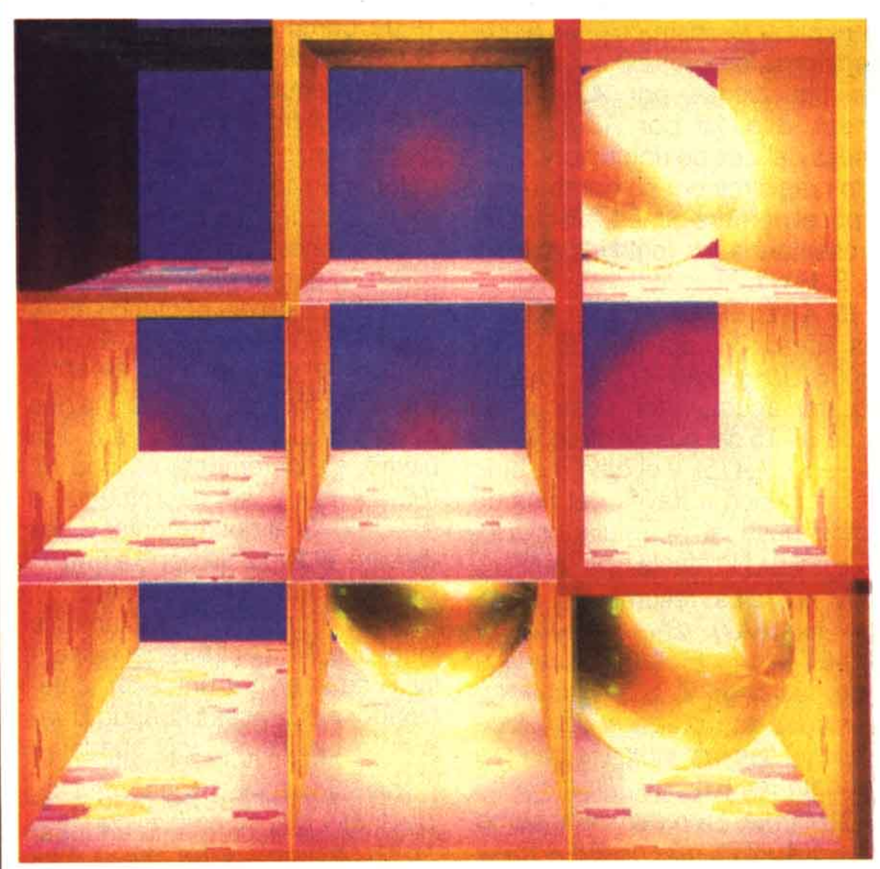
OTIR: Es igual que *OUTI*, pero se repite hasta que *B=0*.

OUTD: Es igual que *OUTI*, pero *HL* se decrementa en vez de incrementarse.

OTDR: Es igual que *OTIR*, pero *HL* se decrementa en lugar de incrementarse.

Y hasta aquí las instrucciones. Vamos ahora con un ejemplo sencillo, que ilustra el funcionamiento de la instrucción *LDIR*. El progra-

código máquina



ma es mixto *BASIC*-Código Máquina. La rutina en máquina rota una serie de posiciones de memoria de forma que la primera pasa al final y las demás se adelantan una posición. El programa en *BASIC* pide una cadena de caracteres y carga cada carácter en una posición de memoria. Luego llama a la rutina en máquina y después imprime los caracteres que lee con *PEEK*, con lo que conseguimos el efecto de un texto rotativo.

```
        ORG B001H
CHAR1:  EQU B021H
CHAR2:  EQU B022H
LONG:   EQU 0000H
        LD DE,CHAR1
        LD A,(DE)
        LD HL,CHAR2
        LD BC,LONG
        LDIR
        LD (DE),A
        RET
```

La directiva de ensamblador *EQU* permite asignar directamente el valor escrito a su derecha a la etiqueta escrita a su izquierda, con lo cual con un pequeño cambio se puede adaptar esta rutina a otros usos. *LONG* no es la longitud total de texto a rotar, sino la longitud menos uno.

CHAR1 apunta al primer carácter, que preservamos en el acumulador con *LD A,(DE)* para que al ejecutar *LDIR* no se pierda este primer carácter. Os recordamos que *LDIR* pasa los caracteres de *(HL)* a *(DE)*, por lo que la primera transferencia será de *(B022H)* a *(B021H)*, sobreescribiendo lo que hubiera antes en *(B021H)*.

Al salir de *LDIR*, *HL* apunta al carácter siguiente al último de la cadena y *DE* apunta al último, por lo cual utilizamos *LD(DE), A* para pasar al final de la cadena el carácter que antes era el primero y que habíamos guardado en el acumulador.

Si hacemos *LONG=0* es porque la longitud del texto se determina por el usuario al ejecutar el programa en *BASIC*, y ese valor menos uno es el que se «pokea» en la línea 100.

El programa en *BASIC* es el siguiente:

```
10 CLEAR 200,&HB000
20 S=0:FOR N=1 TO 14
30 READ D$:D=VAL("&H"+D$):
   S=S+D
40 POKE &HB000+N,D:NEXT N
50 IF S<>1128 THEN PRINT
   "ERROR EN LOS DATA-
S":END
60 DEF USR=&HB001
70 INPUT "TEXTO (MAX. 32)";T$
80 L=LEN(T$):FOR N=1 TO L
90 POKE &HB020+N,ASC
   (MID$(T$,N,1)):NEXT N
100 POKE &HB009,L-1
110 CLS
120 LOCATE 0,0
130 FOR N=1 TO L:PRINT CHR$
    (PEEK(&HB020+N));:NEXT
140 A=USR(0):GOTO 120
150 '
160 DATA 11,21,B0,1A,21,22,B0,
    01
170 DATA 00,00,ED,B0,12,C9
```

Si probáis el ejemplo, veréis que el movimiento resulta un poco irregular; esto es debido a que utilizamos el *BASIC* para imprimir los caracteres. Más adelante veremos una rutina para hacer mensajes rotativos enteramente en código máquina.

Con esto terminamos la parte teórica de las instrucciones del *Z80*. A partir de ahora iremos viendo rutinas útiles aplicadas a los MSX.



Lector de sectores

Los afortunados usuarios que posean unidad de disco también habrán tenido algún que otro problema.

Sobre todo cuando nuestro preciado periférico empieza a hacer cosas extrañas como leer el disco de vez en cuando o dejar de hacerlo repentinamente. Esto puede ser debido al disco en sí. En tal caso, la rutina siguiente permite volcar a impresora el contenido de los sectores del disco.

Como hemos dicho anteriormente, es de suma utilidad para recuperar información de un disco defectuoso o deteriorado.

```
10 REM ******************************
20 REM ***** LECTOR DE SECTORES *****
30 REM ******************************
40 SCREEN 0:COLOR 15.4.4
50 KEY OFF:LOCATE 1.4:PRINT "INTRODUZCA
    UN DISCO Y PULSE UNA TECLA"
60 IF INKEY$="" THEN 60
70 A=DSKF(0)
80 PRINT:PRINT
90 PRINT TAB(6):"HAY":720-(A*2):"SECTORE
    S OCUPADOS"
100 PRINT:PRINT
110 PRINT TAB(5):"1)- DISCO COMPLETO"
120 PRINT
130 PRINT TAB(5):"2)- VARIOS SECTORES"
140 PRINT:PRINT
150 INPUT "SELECCIONA OPCION (1-2) ":A
160 IF A<>1 AND A<>2 THEN 150
170 IF A=2 THEN 240
180 FOR A=0 TO 720
190 NS=A:GOSUB 320
200 NEXT A
210 CLS:LOCATE 9.4:PRINT "PULSA UNA TECL
    A"
220 IF INKEY$="" THEN 220
230 RUN
240 CLS:INPUT "SECTOR DE COMIENZO":SC
250 IF SC<0 OR SC>720 THEN 240
260 PRINT:PRINT
270 INPUT "ULTIMO SECTOR ":SF
280 IF SF<0 OR SF<SC OR SF>720 THEN 270
290 FOR A=SC TO SF
300 NS=A:GOSUB 320
310 NEXT A:GOTO 210
320 A$=DSKI$(0.NS)
330 DI=PEEK(&HF351)+256*PEEK(&HF352)
340 FOR X=DI TO DI+512
350 LPRINT CHR$(PEEK(A)):" ":
360 NEXT X
370 RETURN
```

Caracteres sonoros

Podemos utilizar los vectores de la RAM para interceptar las rutinas del sistema operativo. El siguiente ejemplo muestra como realizar esta operación, interceptando la rutina de impresión de caracteres, de manera que cuando se imprima un caracter en la pantalla suene un BEEP y haga una pequeña pausa antes de imprimir el siguiente. Una vez ejecutado el programa, puede borrar mediante la instrucción NEW.

Para volver a la normalidad basta con hacer POKE 64932,201 y para activarlo de nuevo, POKE 64932,195.

```
10 REM ********************************
20 REM ****** CARACTERES SONOROS ******
30 REM ********************************
40 REM
50 CLEAR 200.56127!
60 FOR A=56128! TO 56144!
70 READ X:POKE A.X
80 NEXT A
90 POKE 64932!.195:POKE 64933!.56128!-25
    6*INT(56128!/256):POKE 64934!.INT(5
    6128!/256)
100 DATA 245.197.213.229.205.192.0.225.2
     09.193.241.118.118.118.118.118.201
```

Paleta de colores

Los usuarios de los nuevos ordenadores MSX de la segunda generación, pueden utilizar esta pequeña rutina para elegir los colores adecuados para sus propios programas. En la pantalla se muestra un cuadrado que se puede cambiar de color utilizando las teclas del cursor. Cuando haya seleccionado el color adecuado pulsaremos la barra espaciadora y aparecerá la instrucción correspondiente al color elegido. Es de suma utilidad, ya que evita tener que ojear el manual cada vez que deseemos un color particular.

```
10 REM ***********************
20 REM * PALETA DE COLORES *
30 REM ***********************
40 REM
50 SCREEN 5
60 LINE (40.40)-(190.190).1.BF
70 REM
80 COLOR=RESTORE:D=STICK(0):
90 IF D=1 THEN R=R+1:IF R>7 THEN R=0
100 IF D=3 THEN V=V+1:IF V>7 THEN V=0
110 IF D=5 THEN B=B+1:IF B>7 THEN B=0
120 IF INKEY$=" " THEN 150
130 COLOR=(1.R.V.B)
140 GOTO 70
150 SCREEN 1:PRINT "COLOR=(1.":R".":V:".
    ":B")"
```

Rincón del lector

MEMORIA LIBRE DE USUARIO

Voy a comprarme un ordenador de 64K y me había decidido ya por uno de los ordenadores MSX, sobre todo por la posibilidad de ampliar su memoria RAM mediante cartuchos. Un amigo me ha informado que con dichos cartuchos de ampliación no aumenta la memoria libre para el usuario. Si esto es cierto, ¿qué utilidad tienen estos cartuchos?

Ramón Martínez
Cádiz

No es cierto que la memoria libre para el usuario no aumente con la conexión de cartuchos de ampliación, si bien la memoria extra no es reconocida por el intérprete BASIC que permite un máximo de 28815 *bytes* libres para el usuario. El resto de la memoria puede utilizarse para guardar datos, pantallas, otros programas o simplemente para buffer de impresora. El *BIOS* (conjunto de rutinas utilizadas por el BASIC) dispone de una serie de rutinas que permiten utilizar la memoria añadida mediante cartuchos.

DIVERSOS PROBLEMAS

Soy un lector de vuestra revista, aunque algunos meses tengo problemas para conseguirla. Me gustaría que me resolviéseis algunos problemas que se me han planteado:

1. ¿Cómo puedo pasar un programa desde un cartucho o una cinta a la unidad de disco?

2. ¿La impresora Philips VW030 tiene retroceso de papel?

3. ¿Dónde puedo adquirir en Granada la revista?

Angel Gómez
Granada.

Los programas de cartucho no pueden pasarse a disco ya que no puede evitarse su autoejecución y no se puede sacar el cartucho con el ordenador conectado. Para pasar los programas de cinta, si estos están en BASIC, basta con cargarlos en memoria con la instrucción CLOAD o LOAD "CAS:" y grabarlo con la instrucción SAVE "A:nombre". Para los programas en código máquina hay que averiguar las direcciones de comienzo, final y dirección de autoejecución, mediante un sencillo programa lector de cabeceras y grabarlo utilizando la instrucción BSAVE "A:nombre", dir. inicial, dir. final, autoejecución.

La información sobre la impresora podrás obtenerla de Philips o cualquiera de sus distribuidores.

Si tienes problemas en conseguir nuestra revista, nada mejor que mandar el cupón de suscripción para asegurarte tu ejemplar todos los meses.

DIFERENCIA ENTRE ROM, RAM Y VRAM

Soy un estudiante y recientemente me he comprado un ordenador CANON V-20 de 64K. Aunque soy principiante tengo gran interés en conocer todas las posibilidades de estas máquinas. Por ello quisiera que me resolvieran algunas dudas que se me han planteado:

1. Diferencia clara para un principiante entre ROM, RAM y VRAM.

2. Al conectar el ordenador dispongo de 28815 bytes libres. Como me han dicho que hay programa que necesitan 32K para poderse ejecutar, quisiera saber si mi aparato los puede utilizar.

3. Hay cartuchos de RAM y ROM, ¿cuál es la diferencia?

José Granero
Murcia

La *ROM* (*Read Only Memory* — Memoria Sólo Lectura) es una memoria a la que el ordenador sólo puede acceder para su lectura, no se puede escribir en ella. Por lo tanto no se puede modificar. La *ROM* es una memoria permanente, es decir, cuando se apaga el ordenador los datos escritos en ella no se destruyen. La memoria *ROM* de los MSX, contiene el sistema operativo del ordenador y el intérprete BASIC.

La *RAM* (*Random Access Memory* — Memoria de Acceso aleatorio) es la memoria destinada a almacenar los programas y datos del usuario. Esta memoria es volátil.

La *VRAM* (Vídeo *RAM*) es una memoria *RAM* destinada a almacenar las imágenes que aparecen en la pantalla en cada momento.

2. Todos los programas de 32K se pueden ejecutar en tu ordendor, puesto que tiene 64K. Los 28815 *bytes* libres, es la cantidad de memoria libre para el BASIC y es la máxima capacidad disponible para programas BASIC en cualquier ordenador.

3. La diferencia entre cartuchos *ROM* y *RAM* es básicamente la misma que las memorias *ROM* Y *RAM*. Normalmente, los cartuchos *ROM* contienen programas que pueden utilizarse con solo conectar el cartucho al ordenador, y los cartuchos *RAM* permiten ampliar la cantidad de memoria para datos o programas.

PARAR UN PROGRAMA EN CODIGO MAQUINA

Desearía que me resolvierais un problema que se me ha planteado. ¿Cómo puedo parar un programa en código máquina mediante las teclas CTRL + STOP?

Oscar Mejía
Santander

Durante la ejecución de un programa en código máquina, el intérprete BASIC está inactivo, por lo que la acción de las teclas CTRL + STOP no tiene ningún efecto. Para detener un programa en código máquina, será necesario que éste compruebe que se encuentren pulsadas dichas teclas y retorne al BASIC en ese caso.

En el *BIOS* existe una rutina encargada de comprobar la pulsación de dichas teclas. Esta rutina comienza en la dirección B7 hexadecimal y una vez llamada, devolverá un 1 en el flag de acarreo si las teclas CTRL + STOP se encuentran pulsadas. Para el correcto funcionamiento de esta rutina, es necesario que las interrupciones se encuentren desabilitadas.